

CS 6210: Advanced Operating Systems

Collaboration versus Cheating

Note: this note is longer than we'd like, but it is important, so please read it.

- Our goal in CS-6210 is to provide you with a great opportunity to learn more about operating systems. Part of that is a series of projects that you will be asked to implement in the class.
- Collaboration is a very good thing. On the other hand, cheating is considered a very serious offense and is vigorously prosecuted. Vigorous prosecution requires that you be advised of the cheating policy of the course before the offending act.
- If you obtain help of any kind, always write the name(s) of your sources in your project report.
- Note: help includes referencing a web page, discussing the project with others, etc. If you copy code, even from another part of the same project, you need to cite it.
- This document describes what is acceptable collaboration versus unacceptable plagiarism for this class. Note that ultimately, this is an essential part of the Georgia Tech Honor Code.

Collaboration

Collaboration is essential to both the learning experience in this class as well as in the real world. Collaboration is working with other people cooperatively to enhance understanding.

We encourage you to:

- Share ideas
- Explain your code to someone to see if they know why it doesn't work.
- Help someone else debug if they've run into a wall.

Ideally, we'd like to see you do this via Piazza, in public threads. That creates transparency and allows other students to also participate in the conversation as well as learn from it.

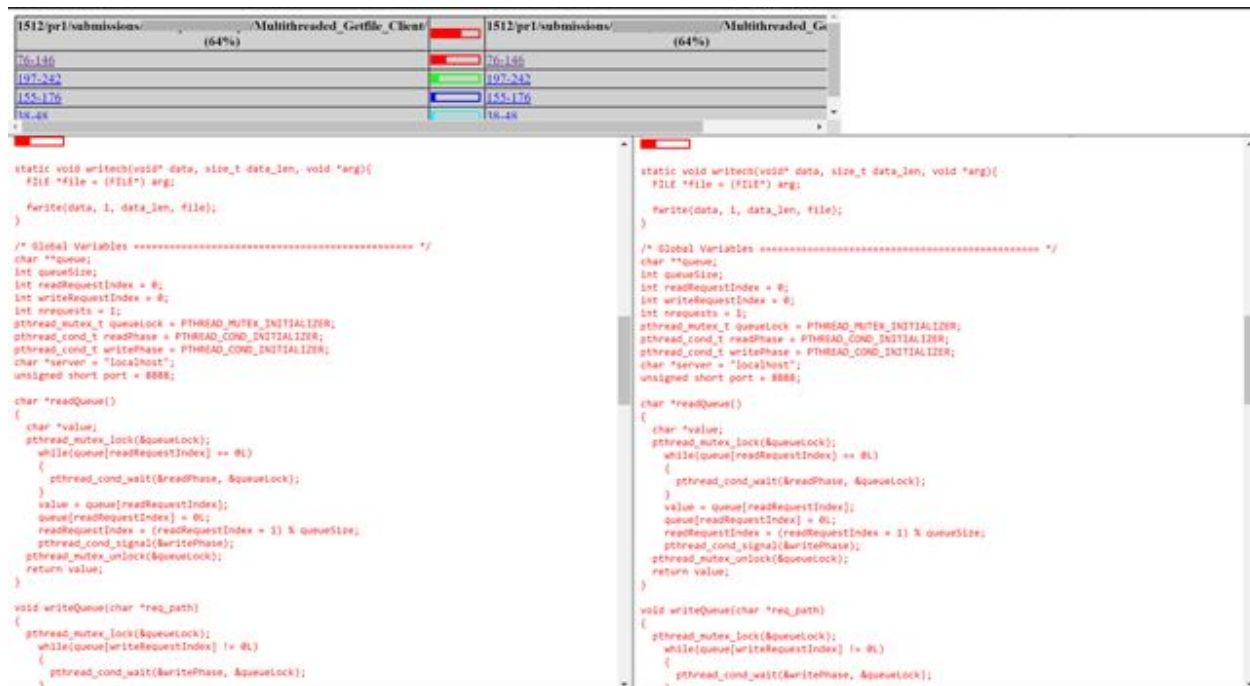
Plagiarism

Plagiarism is using someone else's work instead of doing your own work:

- Never share code or text on the project.
- Never use someone else's code or text in your solutions.
- Never consult project code or text that you find anywhere, such as the Internet

The one exception here is that you may use small snippets of code for commonly performed tasks (e.g., creating a socket, or translating a hostname to an IP address), provided that you properly reference the usage and explain what the code does in your project report. **A snippet is 10 or fewer lines; the maximum number of such snippets you may use is three (3).**

Note: a "snippet" is less than 10 lines of code. If we find you used 11 lines of code once in your project, we aren't likely to say anything. But let me show you a real-world example of this:



```
static void writecb(void* data, size_t data_len, void *arg){
    FILE *file = (FILE*) arg;
    fwrite(data, 1, data_len, file);
}

/* Global Variables ===== */
char **queue;
int queueSize;
int readRequestIndex = 0;
int writeRequestIndex = 0;
int nrequests = 1;
pthread_mutex_t queueLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t readPhase = PTHREAD_COND_INITIALIZER;
pthread_cond_t writePhase = PTHREAD_COND_INITIALIZER;
char *server = "localhost";
unsigned short port = 8888;

char *readQueue()
{
    char *value;
    pthread_mutex_lock(&queueLock);
    while(queue[readRequestIndex] == 0)
    {
        pthread_cond_wait(&readPhase, &queueLock);
    }
    value = queue[readRequestIndex];
    queue[readRequestIndex] = 0;
    readRequestIndex = (readRequestIndex + 1) % queueSize;
    pthread_cond_signal(&writePhase);
    pthread_mutex_unlock(&queueLock);
    return value;
}

void writeQueue(char *req_path)
{
    pthread_mutex_lock(&queueLock);
    while(queue[writeRequestIndex] != 0)
    {
        pthread_cond_wait(&writePhase, &queueLock);
    }
}
```

What do you think the odds are that two **different** students would write exactly the same code? Here's an experiment for you: try rewriting code that you previously have written **without looking at that older code**. Then compare them. If you can't write the same code twice, what do you think the odds are that someone else will.

That image is from one of the tools that we use in this course. It is not the only mechanism that we use, it just happens to be the most well known of them. There's even a public github that discusses techniques for bypassing it.

<https://github.com/genchang1234/How-to-cheat-in-computer-science-101>

Here's the interesting bit of it - if you actually do all the work necessary to escape detection, by the time you're done, you'll have done more work than if you had just done your own work.

Here's one of the best descriptions about cheating (and why every excuse doesn't work) I've ever read: <http://www.cs.ubc.ca/~tmm/courses/cheat.html>

Here's an amazing description of why this matters:

https://secure.apa.uoit.ca/academic_integrity/module1/Module13.html

If in doubt, ask.

Potential Penalties

Sadly, despite our best efforts, we find students cheating. While this comes in a number of forms, we do not tolerate it because it is not only unfair to other students but also to you - our goal is to help you learn the material and cheating does not achieve that goal.

Thus, the worst penalty for cheating is that you deprive yourself of the learning opportunity. In addition to disappointing yourself, you also disappoint those of us who work hard to try and help you learn the material.

Cheating is considered as scholarly misconduct. The actual penalty for cheating may consist of not receiving any points on the assignment, failing the class, or may have more serious consequences to your status as a Georgia Tech student, including expulsion from the program.

This is **not** what we want to do - but we have done it in the past. Nobody wants to face a formal review board hearing, presented with evidence of cheating and forced to defend their actions.

Why do we do it? Because if we don't, course gets a poor reputation and that diminishes the value of the program (and the degree) for the vast majority of students that don't cheat.

For you game theory folks: this is our Nash Equilibrium. We're trying to be transparent about it because we want your Nash Equilibrium to be **not cheating**.

Process

- If, prior to the end of the class, we suspect you of cheating, we will open a private thread on Piazza showing you the information we have regarding our conclusion. This is the same information that we share with OSI when your case is reported. At that point you will be given the option of accepting a zero: either on the portion of the project on which you cheated (if there is only one) or on the entire assignment (if we detect plagiarism on more than one part). This is known as a Faculty Conference Resolution. If you clearly accept the FCR, we report that outcome to OSI and they authorize us to adjust your grade. If it is not clear to us that you have accepted the FCR, we will refer it to OSI and permit them to make the final determination - note that a referral to OSI that is not an FCR means they will decide your grade for the project (the normal penalty is a zero).
- If, at the end of the semester, there is an outstanding issue, you will receive an Incomplete (I) for the class and we will wait until the matter has been resolved by OSI. Note that you cannot graduate with an incomplete.
- **If you have any questions, please ask us.**
- **If you find someone else is cheating, alert us immediately.**
- **If you find you have cheated, come forward.** We will be fair in working it out with you. We're far less forgiving when we have to come to you with our concerns.
- We had to submit reports on ~ 6% of the class last semester. If you can find a repository with code from a prior semester, please know that we have a copy of it because we have code from every semester this class since its inception and the process of checking is now automated. We'd much rather spend the time helping you understand the material than reporting you.
- Here is our recently published paper (the actual conference presentation is March 2, 2019) regarding our activities in this class. <https://dl.acm.org/citation.cfm?id=3287443>
- In addition, while looking at the activity from prior semesters I found it interesting that in most cases people who had to go through the process only a few of them had the penalty (a grade of zero) reduce their final grade. So the only thing worse than cheating is to cheat when it doesn't matter.

Note: we do not review your grade when evaluating your submission for potential plagiarism because it just doesn't matter.

How to avoid cheating

- Do not cut and paste any code
- Do not have someone else's code and your own code open at the same time; read their code. Understand what that code did. Close that code. Open your editor, write your own implementation. I know you think that "it will look the same" but in fact it doesn't.
- Do not submit anyone else's code as your own - the plagiarism policy applies to every submission you make, not just to the one you submit for grading.
- Do not wait until the last minute and then decide you'd rather cheat and hopefully get away with it.
- Do not convince yourself that "my work/another class/etc" had some different policy that you can apply to this course.
- Do not forget to properly cite all your sources.