University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

CSE 220: Systems Programming

1 - *Introduction*

Karthik Dantu
Ethan Blanton
Computer Science and Engineering
University at Buffalo
`kdantu@buffalo.edu`

CSE 220: Systems Programming

- **Instructor:** Karthik Dantu (this section)
  Ethan Blanton (another section)
- **Office Hours:** 12:30-2:00 pm (MW)
- **Course Website**: https://droneslab.github.io/cse220/
- **Discussion Forum**: https://piazza.com/buffalo/fall2019/cse220/resources
- **Syllabus**: https://droneslab.github.io/cse220/lectures/syllabus.pdf
- **Textbooks**:

  Randal E Bryant and David R O'Hallaron. *Computer Science: A Programmer's Perspective.* Third Edition. Pearson. 2016.

  Brian W Kernighan and Dennis Ritchie. *The C Programming Language.* Second Edition. Prentice Hall. 1988.

CSE 220: Objectives

- **Objective**: Understand how hardware (processor, memory, GPU, disks, network) and software (OS, compilers, libraries) come together to execute application programs
- **Benefits of CSE 220**

  Become better programmers

  Identify and eliminate bugs and program bottlenecks efficiently

  Understand and tune program performance

  Stepping stone for other Systems classes in CS and CE

    CSE 421: Operating systems

    CSE 489: Modern Networking Concepts

    CSE 305: Intro to Programming Languages

    CSE 341: Computer Organization

    CSE 321: Realtime and Embedded Systems

    …. and many more

CSE 220: Content

- Hardware
  - Decades of evolution of processor technology
  - Memory and storage have evolved in parallel
  - I/O (including disk and network) dominate interactivity changing the usage model
- Software
  - OS abstracts the hardware and provides a unified interface (system calls)
  - Libraries provide interfaces for commonly used programming fragments (data structures, math operations, user interaction etc.)
- Application programs
  - Sit on top of all the above hardware/software
  - Execute *higher-level commands*
  - Are typically compiled/interpreted by a compiler/interpreter
- This class helps you understand the interaction of application programs with all the above - and hopefully interest you in delving into looking under the hood of a computer program!

CSE 220: Expectations

- Probably your toughest CSE class until now - one of the harder classes in general!
- Attendance - Mandatory!
  - I will not re-do lectures
  - Recitations will also not repeat lectures
  - If you skip lectures, expectation is that you'll catch up on your own
- Labs - at least as important as the lectures!
  - Practice what you learn in class
  - Frequent lab exams to test your understanding
  - Significant portion of the grade
- Assistance - ask for help *early* and *often*
- Meet pre-reqs
  - Some programming experience
  - Understand linked lists and object references

This class elevates expectations from you in terms of your off-class learning. We strongly believe this is the systems-way of thinking – and 220 is designed to inculcate this in you!

CSE 220: Etiquette

- Attend every class and lab !
- Be respectful to instructors, TAs and classmates
- **Adhere strictly** to the academic integrity policy (more in a bit)

Behave as adults and strive to maximize your and your classmates' learning experience in this course.

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

CSE 220: Ways To Fail

- Missing classes and labs
- Start assignments at the last minute
- Not visiting office hrs
- Not asking questions on Piazza
- Waiting until the deadline to submit for the first time
- Cheat!

CSE220: Academic Integrity Guidelines

Academic Integrity

Cheating is submitting any work that you did not perform by yourself as if you did.

CSE220: Academic Integrity Guidelines

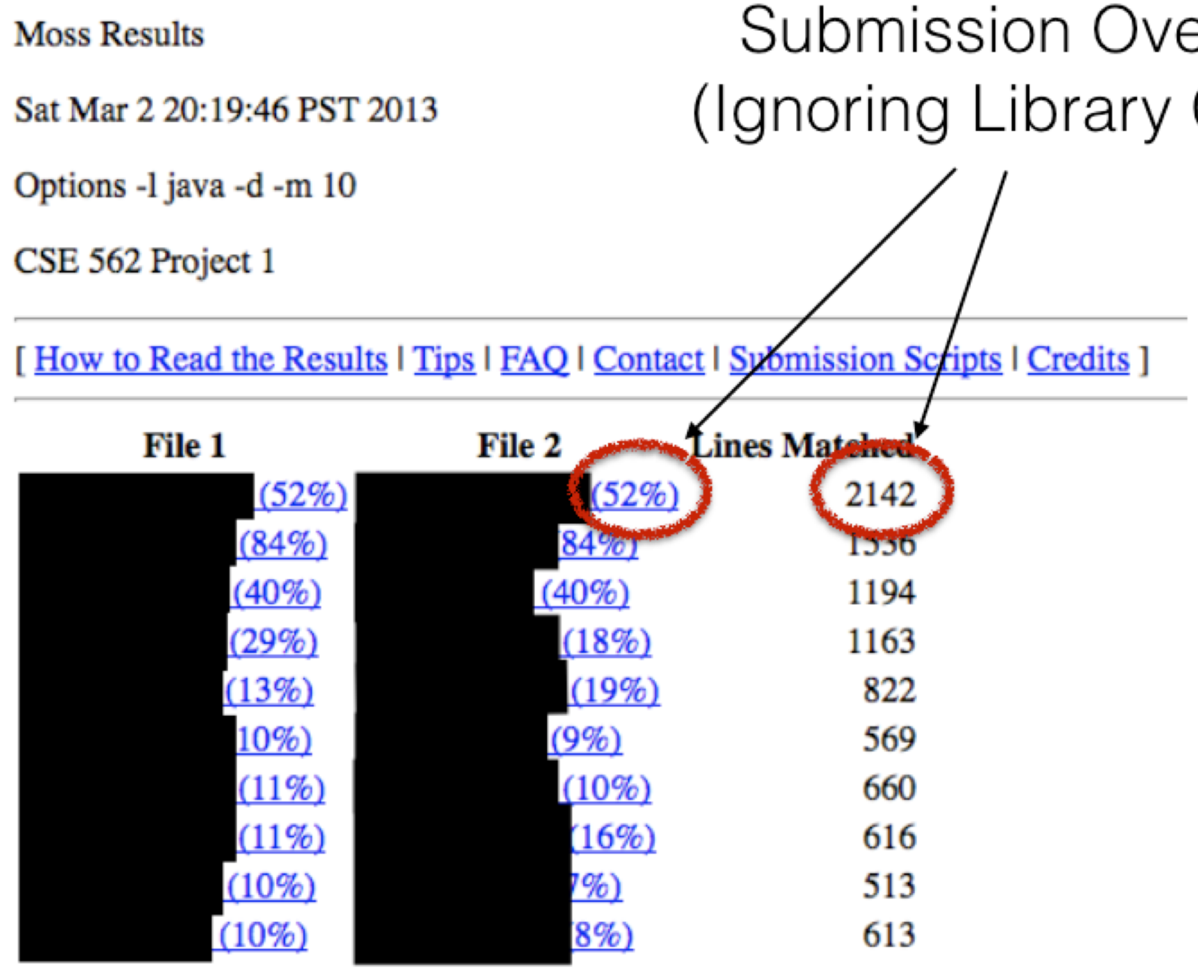
- References (when cited)
  Wikipedia, Wikibooks (or similar): **OK**
- Public Code
  StackExchange/StackOverflow: **Not OK**
- Discussing concepts/ideas with classmates
  "A hash index has O(1) lookups": **OK** (except exams)
- Sharing code or answers with anyone
  "Just look at how I implemented it": **Not OK**
  For-hire code: **NOT OK**

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

CSE 220: Academic Integrity

- We use sophisticated code checkers such as `moss`
- Trust me – it is better than any disguise you can do in short order!
- We also compare with submissions from previous years, as well as publicly available repos

Submission Overlap
(Ignoring Library Code)



Moss Results

Sat Mar 2 20:19:46 PST 2013

Options -l java -d -m 10

CSE 562 Project 1

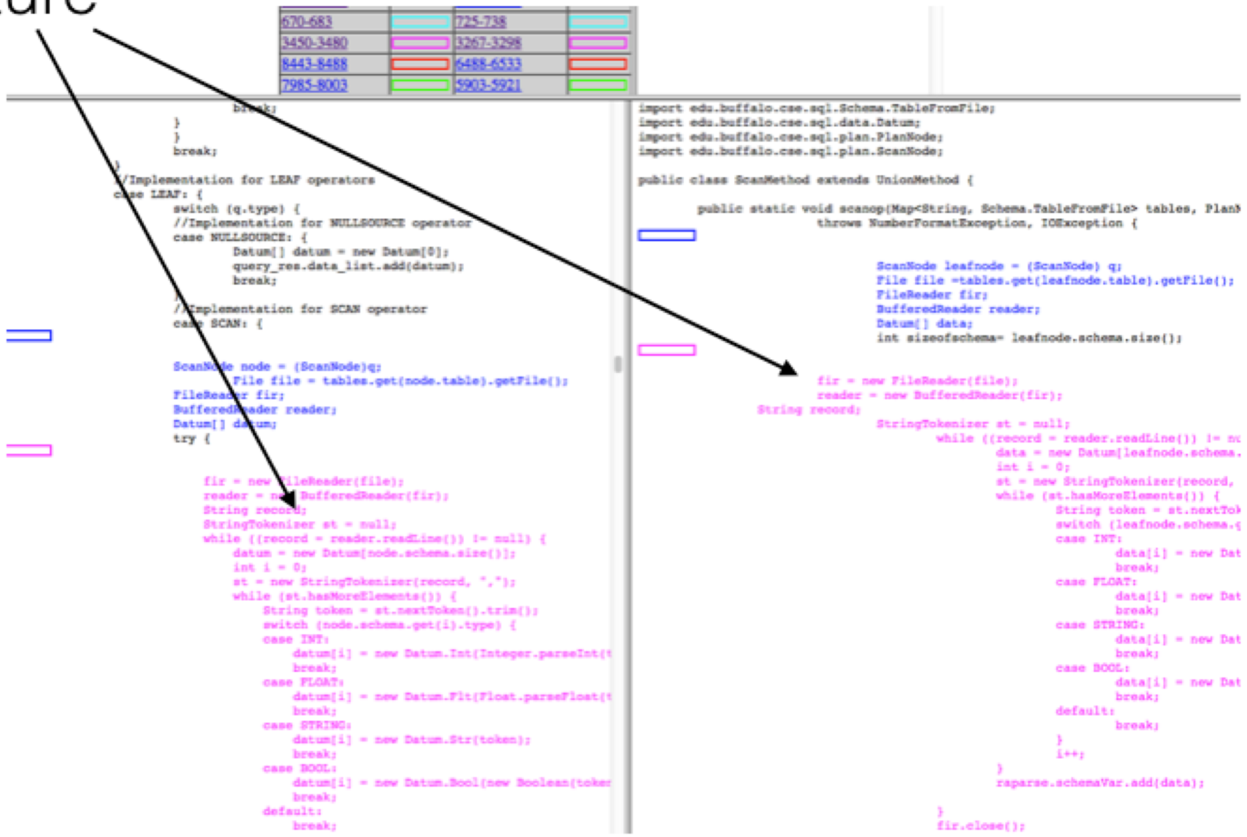
[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
[redacted] (52%)	[redacted] (52%)	2142
[redacted] (84%)	[redacted] 84%)	1536
[redacted] (40%)	[redacted] (40%)	1194
[redacted] (29%)	[redacted] (18%)	1163
[redacted] (13%)	[redacted] (19%)	822
[redacted] 10%)	[redacted] (9%)	569
[redacted] (11%)	[redacted] (10%)	660
[redacted] (11%)	[redacted] (16%)	616
[redacted] (10%)	[redacted] 7%)	513
[redacted] (10%)	[redacted] 8%)	613

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

CSE 220: Academic Integrity

Identical Code Structure



670-683	🟦	725-738	🟦
3450-3480	🟪	3267-3298	🟪
8443-8488	🟥	6488-6533	🟥
7985-8003	🟩	5903-5921	🟩

```
                        break;
                }
                }
                break;
        }
        //Implementation for LEAF operators
        case LEAF: {
                switch (q.type) {
                //Implementation for NULLSOURCE operator
                case NULLSOURCE: {
                        Datum[] datum = new Datum[0];
                        query_res.data_list.add(datum);
                        break;
                }
                //Implementation for SCAN operator
                case SCAN: {


                ScanNode node = (ScanNode)q;
                        File file = tables.get(node.table).getFile();
                FileReader fir;
                BufferedReader reader;
                Datum[] datum;
                try {


                    fir = new FileReader(file);
                    reader = new BufferedReader(fir);
                    String record;
                    StringTokenizer st = null;
                    while ((record = reader.readLine()) != null) {
                        datum = new Datum[node.schema.size()];
                        int i = 0;
                        st = new StringTokenizer(record, ",");
                        while (st.hasMoreElements()) {
                            String token = st.nextToken().trim();
                            switch (node.schema.get(i).type) {
                            case INT:
                                datum[i] = new Datum.Int(Integer.parseInt(t
                                break;
                            case FLOAT:
                                datum[i] = new Datum.Flt(Float.parseFloat(t
                                break;
                            case STRING:
                                datum[i] = new Datum.Str(token);
                                break;
                            case BOOL:
                                datum[i] = new Datum.Bool(new Boolean(toker
                                break;
                            default:
                                break;
```

```
import edu.buffalo.cse.sql.Schema.TableFromFile;
import edu.buffalo.cse.sql.data.Datum;
import edu.buffalo.cse.sql.plan.PlanNode;
import edu.buffalo.cse.sql.plan.ScanNode;

public class ScanMethod extends UnionMethod {

        public static void scanop(Map<String, Schema.TableFromFile> tables, PlanN
                        throws NumberFormatException, IOException {


                                ScanNode leafnode = (ScanNode) q;
                                File file =tables.get(leafnode.table).getFile();
                                FileReader fir;
                                BufferedReader reader;
                                Datum[] data;
                                int sizeofschema= leafnode.schema.size();


                        fir = new FileReader(file);
                        reader = new BufferedReader(fir);
                String record;
                                StringTokenizer st = null;
                                        while ((record = reader.readLine()) != nu
                                                data = new Datum[leafnode.schema.
                                                int i = 0;
                                                st = new StringTokenizer(record,
                                                while (st.hasMoreElements()) {
                                                        String token = st.nextTok
                                                        switch (leafnode.schema.g
                                                        case INT:
                                                                data[i] = new Dat
                                                                break;
                                                        case FLOAT:
                                                                data[i] = new Dat
                                                                break;
                                                        case STRING:
                                                                data[i] = new Dat
                                                                break;
                                                        case BOOL:
                                                                data[i] = new Dat
                                                                break;
                                                        default:
                                                                break;
                                                        }
                                                        i++;
                                                }
                                                raparse.schemaVar.add(data);

                                        }
                                        fir.close();
```

Code in Case Statement

Code in "Operator Class"

CSE 220: Academic Integrity Policy

- First offense
  - Zero on the assignment, lower grade, or an F in the class (instructor judgement)
  - Your name reported to university black list that will follow you through your time at UB
  - If your name is already on black-list, you get an F with note – **you will fail the class and transcript will reflect your cheating**
- Share code, share blame
  - If someone else submits your code as their own, you will be penalized as well

CSE 220: AI Best Policies

- Be careful with your code – including permissions on shared UB filesystems, GitHub, Bitbucket etc.
- **Don't look** at someone else's code!
- Cite liberally
- Check with department[1] and university[2] policies
- Talk to instructors/TAs if you have any questions

[1] https://engineering.buffalo.edu/computer-science-engineering/information-for-students/policies/academic-integrity.html

[2] https://academicintegrity.buffalo.edu/policies.php

Questions/Concerns?

CSE 220: Expectations

- Systems hacking can be fun!
- My best undergrad course was Operating Systems – the sort-of follow up to this course
- You get out of it what you put in; no more, no less !
- If you are willing to put in time, we are more than happy to help
- If you need a better grade, do better work

I hate grade negotiations at the end of the semester – please chat with me through the semester on your progress, but you will get the grade you earned at the end !

CSE 220: Other Logistics

- Re-grading is done only for grading errors
- No incompletes are given in the class
- In principle, no makeup exams will be given except for valid reasons

  Please make sure you talk to me well in advance if you have a valid reason – I will not entertain last minute requests
- **No grades will be changed** for any reason other than grading error

CSE 220: Course Materials

- Follow progress on the course website
- Revise using slides on the course website
- Each class has required and optional readings – read them even if I don't mention it in class
- All discussion/communication will be via Piazza

  Register on Piazza today!

  Make sure you read the posts before you ask a question

  However, if you are not sure about something, just ask!

All important class announcements and materials will be posted on Piazza – you are responsible to follow them!

CSE 220: Platform

- You will work on Linux on x86-64 hardware
- We have a VM Image[1] for you to get started quickly
- You don’t need to use this image
  - But you will be responsible to make sure your submissions are compatible to this image
  - We will not support platform issues on any other platform
- If in doubt, please use this image

[1] https://www.cse.buffalo.edu/~eblanton/misc/vm/

Programming Projects

- Significant portion of your course grade will be projects
  - These are individual projects
  - Projects will be written in C
- To reiterate: projects must run on the course VM image
- We will use GitHub Classroom
  - For assignment distribution
  - For providing assistance
- You must have (or create) a GitHub account
- You are expected to use git and GitHub for development
- If you are not already familiar, learn git
  - [1] Git book: https://git-scm.com/book/en/v2
  - [2] Git tutorial: https://alistapart.com/article/get-started-with-git/
  - [3] Git usage: https://www.google.com/search?q=using+git

Project Assistance

- TAs will be your primary source of help for projects
- To get the most out of the TAs, do:
  - Try the obvious things first
  - Create minimal examples to show problems
- Consult the documentation
- To avoid wasting TA time and failing to get help, don't:
  - Ask for help before you've tried to understand the problem
  - Start at the last minute
- That said, if you've spent many hours to identify the problem and can't, please ask for help

Programming Tools

- We will learn a lot of tools in this class
- You will be expected to use a few tools for this course:
  - The C compiler
  - Make
  - Gdb
  - A programming editor
  - Others …

- We will help you learn these tools

Editors

- We don’t care what editor you use
- Your life will be simpler if you use a programmer’s editor that can do the following:
  - Syntax highlighting
  - Automatic indentation
  - Brace/paranthesis matching
  - Extensibility
- It will be difficult for us to help you if you are not using a reasonable environment
- I personally use emacs – installed on the course VM image
- Other candidates
  - vi
  - Sublime
  - Atom

Project Submission

- All submissions are done through autograder [1]
- Submission rules:
  Submitted within 24 hrs after the deadline: -20%
  Doesn’t count Saturday or Sunday
  Doesn’t count University holidays
- Projects submitted after 24 hrs will not be accepted
- Example: Project is due Friday at 11:59 PM, turned in Monday at 3 pm – 20% penalty
- Example: Project is due Monday at 11:59 PM, turned in Wednesday at 12:15 AM – not accepted

[1] https://autograder.cse.buffalo.edu/

Grading

- Passing this course requires three major conditions:
  - Completion of the AI quiz with perfect score
  - Completion of Lab 01 with perfect score
  - At least a 60% average on all exams and lab exams
- Failure to achieve any of these three points will result in failure in the course
- Your course grade will be calculated per the information in the syllabus

Today’s Assignments

- Immediately:
  Read the Syllabus
  Join the Piazza instance
- By beginning of lab this week:
  Create a GitHub account if you don’t already have one
  Download and install the course VM
- By Friday:
  Complete the AI quiz:
  https://www.cse.buffalo.edu/~eblanton/misc/academic_integrity/ and turn it in on Autograder

Labs

We have labs this week!

Next Class

- Overview of C
- Overview of POSIX API
- Little bit about data types

Required Readings

- Course syllabus
- K&R: 1.1-1.3