INTRODUCTION TO CSC228

Reading:

- none on this topic
- but see the week-by-week handout for readings you should start (now) on upcoming topics

Scenario: You work for Statistics Canada and your job is to write a program that will give government statisticians easy access to the census data, going back over several decades. They want to retrieve data and perform interesting queries.

Scenario: You want to make your first million by writing a web search engine with some amazing new capabilities.

Coming fresh out of csc148, what will be your biggest stumbling blocks?

Problem 1: The data is HUGE; it won't all fit in memory.

Solution: Leave it in a file but bring smaller pieces in memory when needed.

Problem 2: How are you going to perform operations like searching?

The only way we've taught you to read a file is to start at the top and read through to the bottom. This rules out fast algorithms like binary search. With so much data, that's disastrous.

Solution: Adapt clever data structures (like trees), but build them *in the file* rather than in memory. Now we call them "file structures".

A pointer will now reference a location in the file, rather than a location in memory. (We'll learn how to implement file pointers in C.)

Facing another reality

When you follow a pointer in memory, what has to happen?

When you follow a pointer in a file, what has to happen?

Problem 3: Following a file pointer is so slow that we have to be extremely careful to minimize the number of times we do it.

Solution: Use a binary search tree, but make sure it stays balanced.

Think of the census data. $O(\log n)$ used to sound good, but now the constants are so big that we have to care about them.

- What we have to do $O(\log n)$ times is follow a file pointer (plus some faster in-memory work) — and that's a very slow operation.

Plus, if n is huge, $O(\log n)$ itself can be big.

- If your file contains an index of web pages around the world, n could be 20,000,000. What's $\log_2 20,000,000$?

How can we speed things up?

What 228 is all about

This course is about how to manage LARGE amounts of data.

- Understanding how files work
  so that we can better appreciate these issues
- Clever data structures and algorithms
  that try to offer efficiency in the face of these issues
- Programming techniques
  that allow us to implement the clever data structures and algorithms

It's also about how to manage large software projects. More realistic.

- You will work on larger programs.
- You will be more responsible for figuring things out yourself.
- You will be more responsible for making your own design decisions.
- You will work in groups.

Students who effectively manage

- the software design process
  (i.e., have a serious plan for designing, implementing, and testing their code)
- and their team
  (i.e., have a serious plan for who will do what, when, and a strategy for coping when that doesn't happen)

have a huge advantage over everyone else.

ADMINISTRATIVE DETAILS

Reading:

- csc228 Course Information Sheet
- csc228 Course Guide

You

How many ...

- have taken csc209:
- have taken csc258:
- have taken csc238:
- have taken some third year:

How many ...

- are very comfortable in C:
- feel somewhat weak in C:
- are very comfortable in C++:
- feel somewhat weak in C++:

Computers (St George only)

You will be using the CDF lab.

Account name: `a228xxxx`
Password: your student number
Internet address: `@cdf.utoronto.ca`

You can phone the CDF facility from your home computer. See the CDF guide for details.

You are also free to do your assignments on your own computer.

But, warning: you will be required to hand in all assignments electronically, and they must run on the CDF machines.

Plagiarism; Helping each other

The usual.

- In brief: you must submit work that is your own.
- But you *should* get to know other students and work together to learn course material.
- You will have an opportunity for group work on the project.

BASIC CONCEPTS: PHYSICAL FILES, LOGICAL FILES, RECORDS

Reading:

- FZR chapter 1. Warning: 1.4 and 1.5 (on C++ basics) are confusing
- FZR sections 2.1–2.5
- FZR sections 3.1, 3.3, 3.4, and 3.6–3.9
- FZR chapter 4
- FZR sections 5.1 and 5.2

Other homework:

- Review C++ basics from csc270

COSEQUENTIAL PROCESSING

Reading:

- FZR section 8.1–8.3

Algorithm for Master-file Update

```
read the first master file record, m
read the first transaction file record, t

while (at least one file has not been completely read)

    if (m.key > t.key)
       // No master record exists for this transaction.
       if the transaction can be processed
           process it
           read the next transaction file record into t
       else
           log an error
       end if

    else if (m.key < t.key)
       // No transaction exists for this master record.
       print the [probably unchanged] record m
                               to the new master file
       read the next master file record into m

    else both keys are equal
       // Transaction t applies to record m.
       apply transaction to to record m
       read the next transaction file record into t
       // There may be more transactions for m, so
       // don't read the next master record.

    end if
end while
```