

File Structures - CS 321
Fall 2008
Credits: 3 hours

Instructor: David Furcy

Office: Halsey 220

Email: furcyd@uwosh.edu

Phone: 424-1182

Office Hours: MTuWThF 8:00-9:00AM

Feel free to drop in at any other time when my door is open.

Alternatively, to ensure that I have enough time to answer your questions, I strongly encourage you to make an appointment.

Class Meetings: 9:10-10:10AM on Mondays, Wednesdays and Fridays in HS 208

Prerequisites: CompSci-251 and CompSci-271 both with a grade of C or better.

References: There is no required text for this course. However, students will need to refer to the following sources:

- A recent C++ book with good coverage of the IOStream standard library.
- The course web page at: http://www.uwosh.edu/faculty_staff/furcyd/cs321/

Tests:

Exam #1: Wednesday, October 8th, 5:00-7:00PM

Exam #2: Wednesday, November 12th, 5:00-7:00PM

Exam #3: Wednesday, December 10th, 5:00-7:00PM

Exams are scheduled at night to circumvent the one-hour time limit imposed by in-class exams. Please block these time slots at the beginning of the semester. The procedure and criteria of eligibility for making up a missed exam are described below.

If you have special needs (or ABSOLUTELY cannot make it to one of the above night exams), please come and talk to me as soon as possible so I can accommodate your needs right away.

Course Overview: In your data structures course, you learned how to organize and access data stored in main memory. However, main memory (internal storage) is volatile and quite limited in storage capacity relative to the needs of many programs. The use of secondary memory (external storage) addresses both of these limitations. For example, magnetic disks can store hundreds of gigabytes of data permanently. When moving from data structures stored in main memory to file structures stored in secondary memory, one faces a new set of challenges. First, access to secondary memory is typically several orders of magnitude slower than access to primary memory. Second, unlike direct access to primary memory locations, not all file accesses are equal when it comes to delays. We will explain these differences and study their consequences on the design of file structures. We will study the fundamental file structure concepts and show how to apply them in a variety of environments and task constraints.

Course Goals: The goals of this course are for the student to:

- Appreciate the primary role of long access times in shaping the development of file structures via the minimization of disk *seeking* operations.
- Understand the conceptual underpinnings for and the detailed implementation of the most-widely used file structures and file manipulation algorithms.
- Identify the factors that constrain the choice of a file organization for a specific application, for example:
 - the type of accesses (sequential or direct/random),
 - the expected distribution of basic file operations (record additions, modifications, and deletions),
 - the limitations of the physical medium due to its mechanical components (e.g., an average seek time of 8ms), and
 - the constraints imposed by the operating system's file manager or by the system administrator (e.g., the blocking factor).

Learning Outcomes:

- Given the mechanical characteristics of a hard drive, the student will be able to compute the minimum, maximum, and average latencies of file accesses.
- Given the description of a text- or binary-file processing task, the student will be able to implement a solution for it using the language (e.g., C++ or Java) primitives for accessing external memory, and taking advantage of buffering whenever possible.
- Given a maximum RAM capacity and a set of file records stored on disk, the student will be able to sort them according to a given key field using the k-way sort-merge approach, both with and without the replacement-selection algorithm.
- Given a set of records on file with a primary key and a value of the desired B-tree order, the student will be able to construct the B-tree or B+-tree index of the given file by repeatedly applying the two-pass insertion algorithm.
- Given a B-tree index and a key value, the student will be able to delete the entry in the B-tree corresponding to the file record with the given key value.
- Given the description of a multi-level index for an ISAM sorted file, the student will be able to analyze the trade-off between memory requirements and access times for varying levels and types of index (e.g., dense or sparse).
- Given the number of frames in a database buffer and a string of logical references (page numbers), the student will be able to produce the corresponding string of physical references (disk blocks) for a variety of page-replacement algorithms, including the optimal, worst, FIFO, LRU and clock algorithms.
- Given a character set and a list of key strings, the student will be able to construct the corresponding index in the form of either a regular trie or a PATRICIA tree.
- Given a data file and two (or more) key fields, the student will be able to generate a bitmap index and a 2d-tree index for the file.
- Given a sequence of record keys and a collision-handling technique (linear probing, quadratic probing, double hashing, linked method with overflow area, or bucket hashing) for static hashing, the student will be able to compute the number of collisions incurred when inserting the keys into the index.

- Given an extensible hashing-based index, the student will be able to predict the configuration of its directory file and bucket structure resulting from insertions or deletions of index records.
- Given the contents of a file, the student will be able to trace the LZ77, LZ78, and LZW algorithms and to produce the corresponding compressed file contents.
- Given the contents of a file compressed with one of the LZ77, LZ78, and LZW algorithms, the student will be able to decompress the file manually and recover its original contents.
- Given the description of an encryption scheme such as a substitution cipher or a public-key cryptosystem (e.g., RSA), the student will be able to implement its encryption/decryption algorithm and to simulate it on paper.

Topic Coverage: We will cover the following topics:

- Text versus binary files.
- Fundamental file processing operations.
- Magnetic disk I/O.
- Sequential file processing.
- Simple and indexed searches.
- Hashing techniques.
- Buffering.
- External sorting.
- File compression.
- File authentication and encryption.

Course Grading Policy: Your final grade for this course will depend on (between 5 and 7) programming assignments and 3 exams. Each assignment and exam will be graded on a scale from 0 to 100. All assignments will carry the same weight when computing your overall assignment grade. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Assignments	40%
Exam #1	20%
Exam #2	20%
Exam #3	20%

Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
≥ 92	A	≥ 72	C
≥ 89	AB	≥ 69	CD
≥ 82	B	≥ 60	D
≥ 79	BC	< 60	F

I will be glad to discuss any questions you may have about grades. However, make sure to bring them up right away, upon return of each graded assignment or exam. Last minute requests after the final exam will not be entertained.

Attendance and Participation: You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, to have done the assigned reading (when applicable), and to have completed the programming assignments on time.

It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.

On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage** you to **ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email or give me a call to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an assignment is due.

Late Submissions: I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time) after which any submission is considered late, **with no exception**. The penalty for late submissions is computed as follows:

Turned in	Penalty
on due date, after deadline	10%
one day late	20%
two days late	40%
three days late	60%
four days late	80%

Note that submissions that are more than 4 days late will receive zero points. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from the attending physician or a written justification for the extension from the Dean of Students Office. If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester.

Bonus points: Since late submissions are penalized, it is only fair to reward early ones as well. Therefore, if you submit an assignment at least one full day early (and up to four days early), you will earn 3 bonus points for every full 24-hour interval between your submission time and the deadline time. Thus a maximum of 12 bonus points may be earned for each assignment.

Furthermore, there will be other opportunities for students to earn bonus points throughout the semester.

Collaboration versus Cheating: All submissions must be the work of either one or two students, namely the one(s) whose name appears on the submission (yes, you may choose pair-programming for your assignments). While it is acceptable and encouraged to discuss the assignments with others, you must submit your own work. You may not look at or “borrow” any piece of code of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at <http://www.uwosh.edu/dean/conduct.htm> for details.

Final Note: I expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned an A in this course.