- If you add new files, check makefile
- Recvline and sendall: exchange messages through sockets
- Parsing is needed to read the configuration file from the server side
- Use of buf to send and receive messages
- difftime() for performance measurement -> verify the granularity
- Architecture of get/set
  - client sends command to server
  - server handles command, performs operation on its data structure
  - returns response to client (shell)
  - the actual message exchanged is your designed protocol
- Predicates number = 1 per column * max # of cols
- spaces are allowed in predicates
- parser must validate the column type
- all ordering allowed in config file
- only char type allows spaces in the value
- error validation for duplicate predicates -> check if query is valid
- Communication Triad -> Genre, Audience and Purpose
  - Genre defines both the structure of the document and the expectation of the reader
  - Types of genre -> Proposals, Specifications, Instructions oral presentation, written report, email and telephone conversation
- Types of written report -> Proposal, Specifications document, progress report and user manual
- Types of audience -> intended receiver -> general public, technical expert, multiple stakeholders and colleagues
- Reason for communication -> persuade, inform, propose, justify, challenge, reassure, confirm, validate and recommend

High Context Reader	Low Context Reader
Has knowledge of subject	Has limited knowledge of subject
Has knowledge of project	Has limited knowledge of project
Needs little explanation	Needs detailed explanation

- UML -> Unified Modelling Language
- A complex deductive argument is one that challenges three arguments such as claim, justification and evidence
- Client library is the part of the client application. Client library can get, set and query through network from storage server.
- Client library and shell are a part of client application
- Retrieve existing record using storage_get() function
- Insert new record using storage_set() function
- Update a record using storage_set() function
- Delete a record with storage_set() function using NULL value
- Shell and debugging are important to ease debugging.
- Logging is between client library and storage server
- Inner workings for a shell
  - Read input from command line
  - Praise and interpreted input
  - Check input for errors

    - Execute Command
    - char*readline(const char*prompt) -> read input from command line
- Server must be build able by running make server in the src directory
- Client library must be build able by running make libstorage.a in the src directory
- K.I.S.S -> Keep it simple stupid
- Do not over-engineer or gold plate your system
- Design is organizing and planning
- SVN repository -> tracks all modifications to files and directory structure
- SVN's copy-modify-merge model
    - each user connects to repository to retrieve working copy of source tree
    - work simultaneously and independently and modify private copies
    - working copied are merged with svn assists in merging and user may have to resolve conflicts manually
- Recommended Development Cycle
    - Update sources from repository
    - Develop the feature, fix the bug
    - Add new test cases to text new functionality
    - Text with existing and new test cases
    - Commit sources including the new test cases to the repository
- A database management system (DBMS) offers
    - Transaction support
    - Recovery from failure
    - Security and access control
    - Flexibility
    - Management of configuration data
- errno -> returns value to indicate failure
    - typically -1 for NULL pointer
    - need to consult error code stored in errno.h
- Parsing is needed
    - Configuration file
    - Bulk loading of data files
    - Protocol messages -> network
    - Command line arguments (string)
    - Scanning (tokenizing or lexical analysis)
    - analysis of structure and syntax according to a grammar
    - Flex is the scanner generator (open source)
    - YACC is the parser generator
    - YACC is yet another compiler-compiler
    - Lex and YACC work together
    - We use flex (fast Lex) and BISON (GNU YACC)
        - input stream -> Lexical Analyzer -> token stream -> structural analyzer -> Output defined by actions in parser specification
        - Lex
            - Input -> three parts: Definition, rules and user code
            - Definitions define variable and macros
            - Actions are executed when corresponding pattern matches
            - flex picks first rule
            - [xyz] matches either with x or y or z
            - [^a-z] negated character class -> that is anything except those in the bracket

        - {2,5} anything from 2 to 5
    - YACC
        - Definitions, Rules and User Codes
        - Definitions for use in parser code
        - Rules: Grammar for the parser
        - User Code: Code in this part copied to the parser generated by YACC
        - table_list is called a non terminal
        - TABLE & STRING are terminals
    - Benefits of YACC and Lex
        - Faster
        - Easier to change
        - Less error- prone
        - Cost: Learning Curve
- handle_command -> process commands from client library, if get or set or query, extract parameters from command, send back response over socket to client using library functions
- modify read_config() to read the table schemas from config line
- Table Schema: Table name, Number of columns, column names, column types
- Store Records: Key, column values
- Records column name: value pairs are separated by commas
- Column name and value separated by white-space
- Column Value -> integer, string
- max_keys -> number of keys in keys array
- int query -> return the number of records found
- For a record to match a set of predicates the record must match every predicate in the set
- Invalid Parameters -> Columns out of order, Type mismatch, column missing, column name mis spelled, cannot use '>' or '<' in string
- storage_query() -> table and predicates are valid, iterate through each record in the table and return the desired results
- storage_set() -> column names exist in table, columns specified in the right order, column values of right type
- storage_query() -> send predicates from client to server, send keys from server to client
- Developer writes the code and specification
- Debugging is what you do when you know that the program is broken
- Testing is a determined, systematic attempt to break a program
- Testing is about finding bugs
- Test Code -> Code at its boundaries, pre and post conditions, assertions and program defensively
- test pre and post conditions
- use assert.h for pre and post condition test, upon failure aborts the program, should be reserved the for unexpected failures, use at interfaces
- Systematic Testing: Test incrementally, test simple parts first then integrate, know what output to expect, verify conservation properties, compare independent implementations, measure test coverage
- Develop test
    - Utility: Program work for reasonable input
    - Robustness and stress: Test range of operating conditions
    - Reliability
    - Performance
- Test Principles

    - Finding Errors
    - Early and continuously
    - Testing requires creativity and hard work
    - Independent Testers
- Consumers as testers -> record, track, document and prioritize bugs
    - Bugs -> Bugzilla
    - Reproduce Bug and fix bug
- Check is a unit testing framework for C
    - Allows one to think about how interface should be designed
    - provides a documented level of correctness
    - Writing unit tests with writing code
    - first write the test and then the code
- Measure Test Coverage
    - gcov
- Profile Program testing
    - gprof
- Process Examples
    - Browser
    - Email reader
- A Process is a program in execution
    - A unit of execution characterized by a single, sequential thread of execution
    - In the OS are identified by unique by unique Process ID
    - Consists of
        - An executable: Code
        - Associated data: Needed by the program
        - Execution Context: Contents of data registers
- Process Control Block
    - Program Counter
    - CPU registers
    - CPU scheduling information
    - I/O status information
- Context Switch: Switching from one process to another switch is called a context switch
- Time to save and load new processes's execution state is called context-switch time
- Overhead time: System does no useful work while switching
- Concurrency: Simultaneous execution of multiple different processes at the same time
- A process -> i = fork()
- fork() -> Process ID of child process
- Fork() helps to connect multiple clients each with a different server, one with a parent and another with child
- Complications with concurrent server with fork()
    - Caching
    - File-access
    - Zombie Children
    - Signals
- Pros and cons of fork() based design
    - Handles multiple client connections simultaneously
    - file/socket descriptors are shared
    - global variables are carbon copies
    - simple

    - non-trivial to share data between processes
    - requires interprocess communication (IPC) mechanisms: sockets, shared memory, semaphores
- Take-Away Statement -> Purpose of the storage server to provide centralized database for storage of various data. Structured approach -> enables positive utilization of data -> increasing productivity -> higher performance -> better experience in case of residential buildings -> gather experience and acquire skills -> practise software engineering and programming -> learn by failing and then succeeding
- Experience
    - Work as a team
    - Share design documents and code
    - Make design decisions and understand trade-offs
    - Write, communicate, demo and present
    - Network programming with stream sockets
- Compiling and building
    - gcc, make, gdb
- Shared development and revision control
    - svn
- Scanning and parsing
    - Lex and YACC
- Unit Testing
    - Check
- Data Conversion
    - Sed, awk, perl
- Plotting and graphing of results
    - gnuplot
- Get and set combination
        - check r's version against the version stored
            - if there is a mismatch
                - return with ERR_Transaction_ABORT
                - else perform the update
- Thread: a unit of execution, belongs to a process
- Per process items
    - Address space
    - Global Variables
    - Open files
    - Child Processes
- Per Thread items
    - Program Counter
    - Registers
    - Stack
- Processes are largely independent and often complete for resources
- Threads are a part of the same job and actively and closely cooperating
- Threads can access all resources held by a process
- pthread_create() -> creates a thread
- pthread_wait() -> waits for a specific thread
- pthread_exit() -> terminates the calling thread
    - any thread in the process calls exit
    - if a thread is not detached its exit status and thread ID are retained for later

- pthread_yield() -> calling thread passes control voluntarily to another thread
- pthread_ID() -> returns thread ID to caller
- pthread_detach() -> Indicates to system that resources allocated for thread can be reclaimed
- (void*)&connfd -> would be a shared variable
- Race Condition -> Several threads manipulate shared data concurrently. The final value of the data depends upon which thread finishes last.
- To prevent race conditions, threads must be synchronized
- Synchronization -> Semaphores and Mutex
- Lock: Prevent Data Inconsistence due to race conditions
- Mutex -> Mutual Exclusion
- Mutex -> Lock
- Non thread safe code also called non reentrant code
- Deadlocks
  - Code halts as thread may wait indefinitely on locks
    - Due to Programmer error or poorly written code
- Thread creation is more efficient than process
  - Sharing data is fast as threads are a part of the same process
  - Enables concurrent processing of requests from multiple clients
  - Can produce race conditions
  - Keep global variables to minimum
- Single server using select system call to handle concurrent clients
- A transaction is a set of operations against the database that are independent of other concurrently executing sets of operations
  - Examples:
    - ATM transaction
    - Flight booking
  - allows correct recovery from failures
  - keep a database consistent
  - Must be atomic, consistent, isolated and durable (ACID)
- Improve paragraph flow
  - amplify meaning
  - extend meaning
  - aid cohesion
  - orient readers
- Unified Modelling Language
  - Use Case Diagrams
  - State Machine Diagrams
  - Communication Diagrams
  - Timing Diagrams
- Waterfall Model
  - full documentation first and then linear development
- Agile Model
    - Documentation in phases and feedback loops
  - Different Agile Methodologies
    - Adaptive Software Development
    - Scrum
    - Emergent Design
- Adaptive Software Development
  - Speculate

    - Collaborate
    - Learn
- Big Design Upfront
    - Build
    - Plan
    - Revise
- Agile Software Development
    - Individuals and interactions over processes and tools
    - Respond to change over flowing plan
    -