- Programming
    - Top of every file in your program
        - Comment
            - Name, date, class, a#, description of code
    - If you have multiple classes
        - One file per class
    - Variables
        - Use sensible names
        - Types or classes start with a capital
        - Variables lowercase
    - Add sensible comments to code
    - Submit
        - Save files
        - Output that shows functionality
    - Learn debugger
    - All commands are in /bin or /user/bin

- Motivation
    - Most modern applications are multithreaded
    - Threads run within application
    - Multiple tasks with the application can be implemented by separate threads
        - Update display
        - Fetch data
        - Spell checking
        - Answer a network request
    - Process creation is heavy weight while thread creation is light weight
    - Can simplify code increase efficiency
    - Kernels are generally multithreaded

- Multithreaded server architecture
    - Client
        - 1 request
            - Server
                - Creates new threads to service the request
                    - Thread
                - Continues to listening
    - Benefits
        - Responsiveness
        - Resources sharing
        - Economy
        - Scalability

- Multicore programming
    - Multicore or multiprocessor system putting pressure on programmers, challenges include
        - Dividing activities
        - Balance
        - Data splitting
        - Data dependency
        - Testing and debugging
    - Parallelism
        - Implies a system can perform more than one task simultaneously
    - Concurrency
        - Supports more than one task making progress
            - Single processor/core scheduler providing concurrency

- Types of parrallelism
    - Data parallelism
        - Distributes subsets of the same data across multiple cores, same operation on each
    - Task parallelism
        - Distributes threads across cores, each thread performing unique operation
- As # of threads grows, so does architectural

Single and multithreaded processes

- Stack
    - Store return addresses

Amdahl's law

- Identifies performance gains from adding additional cores to an application that has both serial and parallel components
- S is serial position
- N processing cores
    - Speedup<= 1/(S+((1-S)/N)
- That is if application is 75% parallel/ 25% serial moving from 1 to 2 cores results in speedup of 1.6 times
- As n approaches infinity speedup approaches 1/S
- Serial portion of an application has disproportionate effect on performance gained by adding additional cores
- But does the law take into account contemporary multicore systems

User threads and kernel threads

- User threads
    - Management done by user level threads

Multithreading

- Many to one
    - Multiple user threads map to single kernel thread
- One to one
    - Each user threads maps to single kernel thread
- Many to many
    - Multiple user threads map to some number of kernel threads
- Two level model
    - Some threads many to many others one to one