Yen-Ju Tseng

San Diego, CA | tyj850916@gmail.com | +1(858) 729-3110 | www.linkedin.com/in/yenjutseng

EDUCATION

University of California San Diego, Jacobs School of Engineering — **San Diego, USA**
Master of Science in Electrical and Computer Engineering — Sep 2021 – Present
- Coursework: Software Foundations, Operating Systems, Principles/Program Languages

National Tsing Hua University, Wei Kung College — **Hsinchu, Taiwan**
- Coursework: Digital Signal Processing, Introduction to Biomedical Imaging — Sep 2020 – Jan 2021

National Taipei University — **New Taipei City, Taiwan**
Bachelor of Science in Communication Engineering — Sep 2015 – Jun 2019
- Coursework: Data Structure, Advanced Computer Programming, Database System

SKILLS

Programming: C++, Java, Python, Kotlin, JavaScript, HTML/CSS, MATLAB
Tools: Git, Visual Studio, Visual Studio Code

PERSONAL PROJECTS

Relational Database System in C++17 (Github Link) — Apr 2022 – June 2022
- Design and build a working relational database like MySQL using C++17.
- This system involved data interpretation, data manipulation, and data querying. Various software design patterns were used during development. Moreover, this database system is able to execute commands involving the indexes, rows, columns, tables, querying, views, and database.
- The database system is built upon the **MVC(Model-View-Controller)** application design pattern. The "top-level" controller of my program will be an Application class.
- Use Scanning, tokenizing, and parsing to handle user input.
- Use the **chain-of-responsibility** design pattern to handle processing of user provided commands.
- Use the **factory** design pattern to handle statements.
- Use indexes and LRU (Least recently used cache) Cache to improve this database system.

Nachos Operating System Implementation in Java (Github Link) — Sep 2022 – Nov 2022
- Implement and test multiple tasks in Nachos operating system.
- Complete the implementation of the **Alarm class**, including **waitUntil(x)**, **timerInterrupt()**, and **cancel().**
- Complete the implementation of **KThread.join()**.
- Implement **condition variables** using **interrupt disable and restore** to provide atomicity, including **sleep(), wake(), wakeAll(), and sleepFor(long x).**
- Implement the **Rendezvous** class to provide a mechanism for threads to exchange values, using **locks** and **condition variables** to manage **concurrency**.
- Implement the file system calls **create**, **open**, **read**, **write**, **close**, **unlink**, **exec**, **join**, **exit** and **halt**.
- Implement support for **multiprogramming** by managing the allocation of pages of physical memory so that different processes do not overlap in their memory usage.
- Implement **Demand Paging, Lazy Loading, and Page Pinning.**

Game Checkers in C++ — Feb 2022 – Mar 2022
- Based on the rules of checkers, using a programming principle of **inversion of control (IoC)** to plug in two **self-designed checker-bots** which compete in a Game of Checkers to the Game framework and visualize the checkerboard on each step to see the current game status.

Non-compressing Archival Storage System in C++ — Jan 2022 – Feb 2022
- Create a new Archive file or open a pre-existing Archive file which will read/write documents as **sequence of fixed size blocks** in binary mode using C++ **fstream**, **ofstream**, and **ifstream**.
- Implement **6** major **actions**: **add**, **extract**, **remove**(permanently), **list**, **dump**, and **compact**.
- Apply **Observer Pattern** to the Archive to register one or more "observers" of the Archive. Observers will be notified when the Archive performs an action (add, extract, etc.).

Build Our Own String Class in C++ (Without Using std::string) — Dec 2021 – Jan 2021
- Build our own string class with **Aspect-Oriented Programming** and use **Adapter Pattern** to make an adapter so that we can handle both String and const char* in only one function when we try to implement those operations for string, such as append, insert, replace, remove, and find.