# CS 3411 Systems Programming

Department of Computer Science
Michigan Technological University

Introduction

# Operating Systems

- We all use operating systems
- Operating systems interact with devices directly
- Provides programmer friendly interface by:
  - Masking low level hardware interface
  - Supplying abstractions
  - Exporting a *system call* interface for user interaction with system resources
    - Different for each operating system!

# Operating Systems

- Operating systems also protect resources
  - Data of one user from other users
  - Memory of one program from another
  - Keeps the processor from being monopolized

# What is Systems Programming?

- Not a very well defined term!
- Part systems administration
- Includes:
  - Creating and maintaining a platform for users
  - Use the system call interface
  - Code that operates in a privileged mode

# Course Topics

- Review: C vs. C++
- Unix file system interface
- Processes
- Linking, Libraries
- Interprocess communication
  - Signals, pipes, sockets
- Terminal I/O
- Shell Programming
- Additional tools if time allows!

# Course Text

- There is no REQUIRED course text
- We'll be using slides and notes during class
- References:
  - Advanced Programming in the UNIX Environment, W. Richard Stevens (Recommended text book).
  - C: A Reference Manual, (5th ed.), Harbison and Steele
  - UNIX Network Programming Volume 1, Networking APIs: Sockets and XTI, Second Edition, W. Richard Stevens

# Grading

- Programs (5) - 60%
  - 5 Slip days (cumulative for all assignments)
  - After slip days used, 20% per day (including Saturday and Sunday)
  - All submissions on Canvas!
- Tests (2) - 30%
- Homework - 10%
- No final exam

# Programming Expectations

- Work independently!
  - Don't show your code to anyone
  - Don't look at code from anyone (including on the Internet!)
  - Can have 'empty hands' discussions
- No copying code from Web unless explicitly stated
- Make an effort to design and debug your own code!
- Read manuals!

# Programming Expectations

- For program assignments to get full points, it must:
  - Perform specified function correctly
  - Always terminate normally (except on certain signals)
    - Program is responsible for checking the sanity of input!
  - Avoid internal errors, e.g, memory leaks, buffer overflows, etc.
  - Reasonably efficient
  - Well documented and well formed
- Unless otherwise specified, assignments must be done in C and will be graded on a Linux system
  - You may use your own machine for development, but make sure your code runs on the lab machines!