# CSE13S notes
## Week of 1/30/23

Assignment 5: Schmidt-Samoa cryptography

- GMP has a function to generate random numbers
- Write utility functions - small ones can still be efficient since compiler will just paste the small function where you've called it
- Miller-prime works better the more iterations of it you go through
- All about the algorithm - code in pure C first before using GMP

Compilers!! (Innards of language translators)

- Simply put, a program that maps an input language to an output language (much easier to do programming language -> assembly but much more difficult to translate human language)
- Roles:
  - Convert high-level source code to machine language
- In the good old days:
  - Computers were programming in binary
    - Hexidecimal
    - Octal
  - It was hard
    - If you hated assembly in cse12, well this was much harder

Why need language translators?

- Computers are a balanced mix of software and hardware
- Hardware functions are controlled by a compatible software program

Compilation process of a C program

- The compiling process is a sequence of four phases. The input for each phase is the output of the previous
  - Source code -> preprocessor -> compiler -> assembler -> linker - exe
- The pre-processor
  - First thing that runs, usually removes comments
  - Pastes in included files and standard library headers
- The compiler
  - The compiler converts hello.i from the preprocessor into assembly code in 7 steps:
    - Lexical
    - Syntax
    - Translation

        - Optimization
        - Storage assignment
        - Code generation
        - Assembly phase
    - Lexical phase breaks down source code into lexemes (see slides)
- The Assembler
    - Assembly code to machine code (binary) is very simple 1 to 1 translation
    - Can perform conversions is 2 ways - (see slides)
        - Forward referencing happens in one or two passes (see slides)
        -
- The linkers
    - Links hello.o along with any other object files and libraries
    - Ensures all dependencies are properly resolved
        - Linker error thrown if definitions to something that cannot be referenced
    - Merges everything into one executable file
        - In this case, hello.exe
- Loaders
    - Lays in the operating system
    - Ensures program and necessary libraries are placed in RAM to prepare them for execution
    - The loader has 4 basic functions
        - Allocation: allocate space in memory for program
        - linking - resolve symbolic references between programs
        - Relocation - fix all dependent locations and point them to the newly allocated space
        - Loading - place machine code and data directly into the processor
- Memory layout
    - See slides for memory layout in unix
- Compilers vs interpreters
    - Compiler
        - A translator that takes a high-level programming language, goes through a sequence of translations and output an executable
        - Translates entire program/code at once
    - Interpreter
        - Directly executes code without needing to compile (python interpreter, ghci (haskell))
        - Translates program one line at a time
    - Which is faster, the compiler or an interpreter? Generally, the compiler

Interoperability

- Code should be interoperable between different environments (as much as is plausible)

Files

Long-term information storage

- We want to store large amounts of data
    - The definition of large has changed by 6 orders of magnitude over the years
- Information stored must survive the termination of the process using it
    - Memory (DRAM) does not survive turning the computer off
    - This is why you don't kill the VM to close it, it corrupts data because all memory is gone
- Files are accessed using names, memory is accessed using addresses (pointers/variables)

File names

- Our computers have millions of files
    - Data center has billions of files
    - The internet has trillions of files
- How do you uniquely specify which file you want?
- How do you find that file among millions of files?
- Files have base names and extensions (base.extension)
    - In some OSs, the extension is a separate entity
- In UNIX, the extension is only used by convention

Files may havei internal structure

- .EXE have headers containing information about the file, as well as text (code), data, relocation bits, and symbol table
- Archive files have headers and object modules

File Access

- Sequential access
    - Read all bytes/records from the beginning
    - Cannot jump around, may rewind or back up
    - Convenient when medium was magnetic tape
- Randoma access
    - bytes/records read in any order
    - Essential for data base systems
    - Read can be:

        - Move file marker (seek), then read or…
        - Read and then move file marker
    - File attributes
        - Files have many attributes that can be put in a file (in windows and mac)
        - UNIX also has same attributes, but does not care about certain attributes
    - File operations
        - Create, delete, open, close, read, write
        - Append, seek, get attributes, set attributes
        - Rename

Directories

- Naming is better than using numbers, but still limited
- Humans like to group things together for convenience
    - We use hierarchy to manage complexity
- File system allow this to be done with directories
    - You may call them folders
- Grouping makes it easier to:
    - Find files in the first place: remember the enclosing directories for the file,
    - Locate related files,
    - Determine which files are related
- Directory operations
    - Create, delete, opendir, closedir
    - Readdir, rename, link, unlink

2/17/23 notes

Processes

- What is a process? *A running program*
- code , data, and stack have its own address space
- Program state (what the state of the program is right now)
    - CPU registers
    - Program counter (current location in the code)
    - Stack pointer
- Only one process can be running in a single CPU core at any given time
    - Multi-core CPUs can support multiple processes

What is an Address space?

- Program execute code
    - Each instruction has an address
- Programs access data

    - Each byte of data also has an address
- We would like to think that our program is the only program executing on the computer
    - But we would be wrong
- See slides

How is that accomplished (what is that?)

- The loader could relocate the instructions by address a base address to each one
- There could be registers that point to the first byte of the program’s memory (base register) that is addred to every memory access
- The processor and operating system could provide virtual memory

In an ideal world…

- The ideal world has memory that is
    - Very large, very fast, and non-volatile
    - But we don’t live in an ideal world
- The real world has memory that is (PICK 2):
    - Large
    - Fast
    - Affordable
- The goal of memory management is to reach for an ideal world as much as possible

Memory hierarchy

- What is the memory hierarchy?
    - Different levels of memory
    - Some are small & fast
    - Others are large & slow
- What levels are usually included?
    - Cache: small amount of fast, expensive memory
        - L1 (level 1) cache: usually on the CPU chip
        - L2: may be on or off chip
        - L3 cache: off-chip, made of SRAM
    - Main memory: medium-speed, medium price memory (DRAM)
    - Disk: many gigs of slow, cheap, non-volatile storage
- Memory manager handles the memory hierarchy

Basic memory management

- Components include
  - Operating system (perhaps with device drivers)
- Fixed partitions: multiple programs
  - Fixed memory partitions created, where memory is divide into fixed spaces
  - Assign a process to a space when it's free
- Mechanisms
  - Separate input queues for each partition
  - Single input queue: better ability to optimize CPU usage
- Multiprocessing works because I/O takes a long long time
  - Most programs really are waiting for I/O, so cpu can work on other programs

Memory and multiprogramming

- See slides

Base and limit registers

- See slides

Allocating memory

- Search through region list to find a large enough space
- Suppose there are several choices: which one to use?
  - First fit: the first suitable hole on the list
  - Next fit: the first suitable after the previously allocated hole
  - Best fit: the smallest hole that is larger than the desired region (wastes least space?)
    - Best is worst because of the small, unusable memory fragments that get left in the memory space
  - Worst fit: the largest hole that is available

Freeing memory

- Allocated structures must be updated when memory is freed
- Easy with bitmaps: just set the appropriate bits in the bitmap
- Linked lists: modify adjacent elements as needed
  - Merge adjacent free regions into a single region
  - May involve merging two regions with the just-freed area

Buddy allocation

- Allocates memory into powers of 2
  - Partitions in half until a suitably small region is found for an object
- Split larger chunks to create 2 smaller regions

Virtual memory

- Basic idea: allow the OS to hand out more memory than exists on the system

- Keep recently used stuff in physical memory
- Move less recently used stuff to disk
- Keep all of this hidden from processes
    - Processes still see an address space from 0-max_address
    - Movement of information to and from disk handled by the OS without process help
- Virtual memory (VM) especially helpful in multiprogrammed systems
    - CPU schedules process B while process A waits for its memory to be retrieved from the disk

Virtual and physical addresses

- Program uses virtual addresses
    - Addresses local to the process
    - Hardware translates virtual address to physical address
- Translation done by the memory management unit
    - Usually on the same chip as the CPU
    - Only physical addresses leave the CPU/MMU chip
- Physical memory indexed by physical addresses
- Virtual memory can extend anywhere on physical memory, so it is better to use (prevents small unusable memory spaces)

What's in a page table entry?

- Each entry in the table entry?
    - Protection: who can read/modify that piece of memory
    - Dirty (modified) bit: set if data in the page has been modified
    - Reference bit:
    - Valid bit:
- See slides

The process model

- Conceptual model of multiprogramming four programs
    - 4 independent processes
    - Processes run sequentially
- Only one proess active at any time

When is a process created?

- Processes can be created in two ways:
    - System initialization: one or more processes created when the OS starts up
    - Execution of a process creation system call: something explicitly asks for a new process
- System call comes from (see slides)

When do processes end?

- Conditions that terminate processes that do

Processes also have a process table coutner

-