Jack Operating System

B03902082 資工四 江懿友

I implemented project #12 of the textbook for my final project. In this project I implemented the Jack OS in Jack language. The Jack OS is eight classes that serves as a standard library / API that controls the computer hardware. The eight classes, Memory, Array, Math, String, Screen, Keyboard, Sys, Output, provides functionalities of memory allocation, array creation, arithmetic operations, string manipulation, I/O device operations (Screen/Keyboard), system functions (init / fini) and text output (to screen).
Following is descriptions of how I implemented each class.

Memory

The memory allocating class allocates memory from the heap segment, which ranges from address 2048 to 16384. I implemented a memory allocator that works like a simplified dl-malloc (glibc's memory allocator).

deAlloc()

Memory.deAlloc() inserts the freed memory chunk into a linked list structure "unsorted_bin".

alloc()

Memory.alloc() first search if there is any memory chunk that is large enough in unsorted_bin. If there is any, then return the chunk and remove it from unsorted_bin, else allocate a new chunk from heap segment.

By this way freed memory chunks can be reallocated, however I didn't implement freed chunk defragmentation, so memory fragmentation could happen if there is a lot of alloc() / deAlloc() happening.

Array

Array is just a wrapper class that calls the underlying Memory class.

Math

multiply()

I implemented the shift-and-add algorithm to do fast multiplication.

divide()

I implemented the divide-and-conquer algorithm mentioned in lecture.

sqrt()

I implemented the binary-search algorithm mentioned in lecture slides.

String

intValue()

I implemented the time-by-10-and-add-next-digit algorithm.

setInt()

I implemented the modulos-by-10 algorithm.

Screen

drawPixel()

This is just a bunch of bitwise operations to set the pixel's color bit.

drawLine()

Implemented the vector drawing algorithm in lecture slides.

drawRectangle()

Implemented with a double for-loop.

drawCircle()

Use the distance equation to decide if a pixel is within the circle's radius.

Keyboard

keyPressed()

This just returns the value of MEM[24576].

readChar()

This is just a infinite while-loop that calls keyPressed() until a keystroke is detected and released. Also calls Output.printChar() to print the detected keystroke.

readLine()

Calls readChar() multiple times until newline character is detected. Stores the characters in a string buffer at the same time. Returns the read line of string.

readInt()

Calls readLine() and calls String.intValue() and then return the integer.

Sys

init()

Calls Memory.init(), Math.init(), Screen.init(), Output.init(), Keyboard.init(), Main.main(), Sys.halt() in this order.

halt()

Just a infinite while-loop that "halts" the execution flow.

wait()

For each wait duration, execute 500 iterations of empty while-loop. 500 is just an empirical number that's tested on my computer.

error()

Displays error message with Output.printChar() and Output.printInt(), and then calls Sys.halt().

Output

initMap()

Had to calculate the bitmap of character 'A' and fill it in.

printChar()

Calls Screen.drawPixel() to draw the character according to the bitmap returned by Output.getMap().