Resource Manager Simulator Design Document

By: Ryan Abel

# Introduction

The purpose of this assignment was to demonstrate exceptional knowledge of resource management, in particular detecting deadlocks. The program requires an input file that specifies different actions taken by processes related to resources. The file is processed using javascript and displayed in a web browser.

# Implementation

## General Implementation Notes

This implementation was written using javascript. The graphing library used for implementation is Dracula.js. This library specializes in creating and attaching nodes. The web page consists of a simple step button. The button will read the contents of a file and determine certain actions to take. This solution was broken up into several files, each of which has it’s own role in producing output.

The main feature of this program is to determine deadlocks. The technique I took to determine deadlock was to topologically sort the graph every time an action is fired does this. If the graph cannot be topologically sorted, it must contain a cycle and therefore a deadlock.

## Presenter.js

The presenter is responsible for using data retrieved from all the helper files to set data on the view. It has the responsibility of firing actions based on events from the screen. It delegates it’s tasks to the other files and then gathers return information to update the view.

## File\_parser.js

The file parser is responsible for parsing a single line in the file. The file parser attempts to return a hash containing the process, the action, and the resource. There is also a function for parsing a header that currently does nothing, but may be used to display the graph’s initial state.

## File\_reader.js

The file reader is responsible from reading a file in from disk. The file reader also keeps track of the current line being viewed in the file and the total number of lines in the file.

## Process.js

The process file holds all information pertaining to a process. It is responsible for creating a process, attaching a resource to a process, finding a process, and determining if a process already exists.

## Resource.js

The resource file holds all information pertaining to a resource. It is responsible for creating a resource, attaching a process to a resource, finding a resource, determining if a resource exists, and holding a queue of processes that want ownership of the resource.

## Graph.js

The graph file is responsible for everything related to graphing. It can add a node, add an edge, show the graph, detect cycles, and remove edges and nodes. It detects cycles using a topological sorting algorithm. This algorithm attempts to sort the graph by removing every edge on the graph. If after the sort an edge still exists, there must be a cycle in the graph.

## Dracula Graphing Library

I used an external library for the GUI work involved in this project. Dracula.js allows the creation and displaying of a graph.