CS3243 OS Project

By: Troy Jarvis, William Pegg and Sha Lou

Table of Contents

[Introduction 3](#_Toc319354232)

[Statement and Purpose 3](#_Toc319354233)

[System Architecture and Design 3](#_Toc319354234)

[Diagram of System 3](#_Toc319354235)

[Data flow 4](#_Toc319354236)

[Code and Compilation 4](#_Toc319354237)

[Modules 4](#_Toc319354238)

[Driver 4](#_Toc319354239)

[File Loader 4](#_Toc319354240)

[Execution 5](#_Toc319354241)

[Data Collection and Analysis 5](#_Toc319354242)

[Conclusion 5](#_Toc319354243)

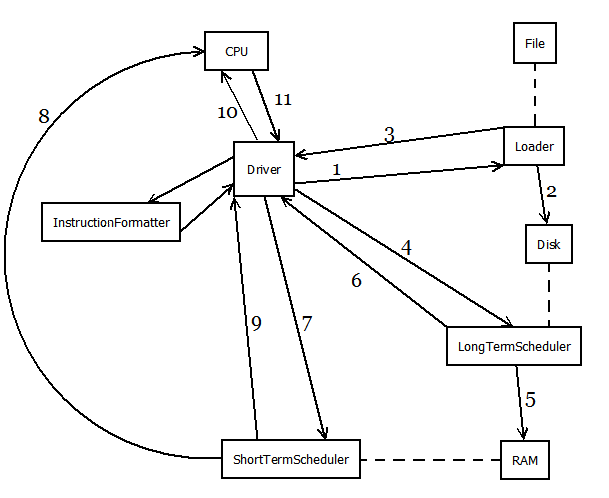
# Introduction

Our project is designed to simulate an Operating System and all of its essential functions. This Simulation includes a Driver, Long Term Scheduler, Short Term Scheduler, Loader, Disk, CPU, RAM, PCB, Instruction Formatter, and File Loader.

## Statement and Purpose

# System Architecture and Design

## Diagram of System



## Data flow

When the system begins the Driver calls the Loader to pull in the processes from a file. The Loader then takes the data and loads into the PCB located on the Disk. The Loader then notifies the Driver that it has completed loading the processes onto the Disk. The Driver then calls the Long-Term Scheduler to move the processes into RAM and then notifies the Driver when it RAM is full and it cannot successfully load another job into it. The driver will then call the Short-Term Schedule to take the processes from RAM and put them onto the CPU to be processed. The Short-Term Scheduler will then notify the Driver that is has completed moving the processes onto the CPU. The Driver will then call the CPU to run the processes. The CPU will notify the Driver once it has completed processing the processes. If there are still processes that need to be run the Driver will call the Long-Term Scheduler to move more processes into RAM and the process will continue until all processes have been run.

# Code and Compilation

## Modules

### Driver

The Driver is essential to the entire operation of the OS simulation. It imports, instantiates and starts the other classes needed to run the simulation. After the Driver has imported all of the different classes, it will create the different queues needed for the processing of the processes (new, waiting, ready, running and terminated). The Driver then creates an instance of the File Loader class which reads in the data from the given file. Next the Driver calls the function displayPidTable() which displays all of the processes that are loaded into the PCB. The next step is for the driver to initialize the jobs by putting them all in the new queue by calling the longTermScheduler.loadProcesses() function. The Driver then calls the Long-Term Scheduler to move to processes into RAM. After the processes have been moved to RAM Dump(), the Driver will call for a dump of the contents of RAM into an output file. The Driver calls the diskDump() which will dump all of the contents of the disk to an output file. The **PUT MORE ABOUT THE DRIVER**

### File Loader

The File Loader only has a few functions consisting of a constructor, a destructor and a function called read\_file, of Boolean type, which takes in a parameter of type PCB and will read the data file provided for the project and load it into the PCB. After the file\_loader has opened the appropriate files it can begin reading in the data and parsing it so that it can be loaded into the PCB. First, it will check to see if the line that was taken in is a header line for the instruction set of the process. If it is header line for the instruction set for the process, it will parse out the ‘/’, ‘JOB’, and ‘ ‘ characters so that it will be able to load the process ID, size and priority of the process into the PCB. Next it will check to see if the line is a header line for the data of the process. If is it, it will parse out the ‘/’, ‘DATA’, and ‘ ‘ characters and set the buffers and the disk address for the coming data into the PCB. After the each data line has been read in it will output the size of the buffers to the screen. Next it will check to see if the line starts with a ‘0x’, if it does this is telling the OS that the coming data is in a hexadecimal format which is not necessary for the actual computation on the processes. The function then will strip of the ‘0x’ from the line and insert the data into the PCB and the disk. The function will then check to see if the line is a line saying notifying the OS that it has come to the end of the processes information and will notify the loader to stop loading into the PCB. It will then dump the contents on the of the disk into two different text files, outfile.txt and outfile\_readline.txt, to be sure that the data matches and was input into the system correctly.

### PCB

The PCB file is the file that will store all of the processes, their instruction and their data.

# Execution

# Data Collection and Analysis

# Conclusion