

# CPSC 410

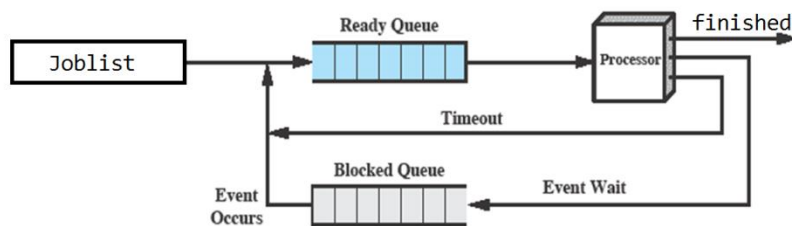
## Project 2

### Motivation: Topics covered by this project;

- Queuing and the processor
- Chapter 3 in Stallings

### Overview

You are to simulate the process queuing model shown from figure 3.8 in your text. Please read the course text and thoroughly understand how this model works before starting this project.



I have provided much of the infrastructure, you fill in the rest. This project builds off of project 1. Note that it runs in a single thread, so you don't have to worry about the complexity associated with multiple threads.

### Teams

Please work 2 to a team. Please include a description of what each team member did as part of your submission. Percentage responsibility for the final product is fine.

## Overview

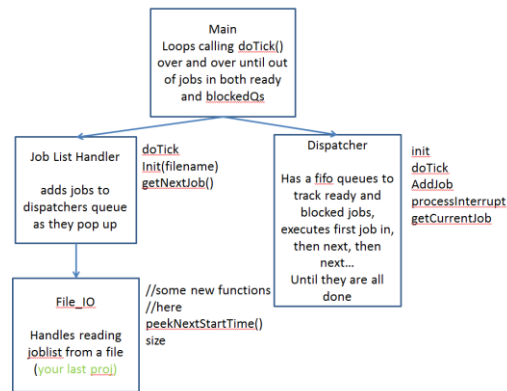


Figure 1 Module interaction

### File\_io (given to you)

This set of functions duplicates what you did in project 1. I changed the name of the structure from process\_stats to PCB to conform to the notion of a process control block. Note the 2 extra functions;

```
//returns when the next job starts without altering the container
int peekNextStartTime();
```

```
//return the number of elements
//in the container
int size();
```

You are given a file that has an unknown number of rows, and **4 columns of integers**. Assume there are no malformed rows (ie != 4 columns). It looks like the following.

```
1,10,7,0
2,5,4,0
3,3,10,1
5,15,12,1
6,0,4,0
7,7,10,1
8,1,19,1
9,1,6,1
```

This file can have any name, but I will call it testdata.txt for the purposes of this document. The first column is process\_number, second is start\_time, third is cpu\_time and the fourth is io\_time.

File\_IO provides functions to load this data into a vector and sort it.

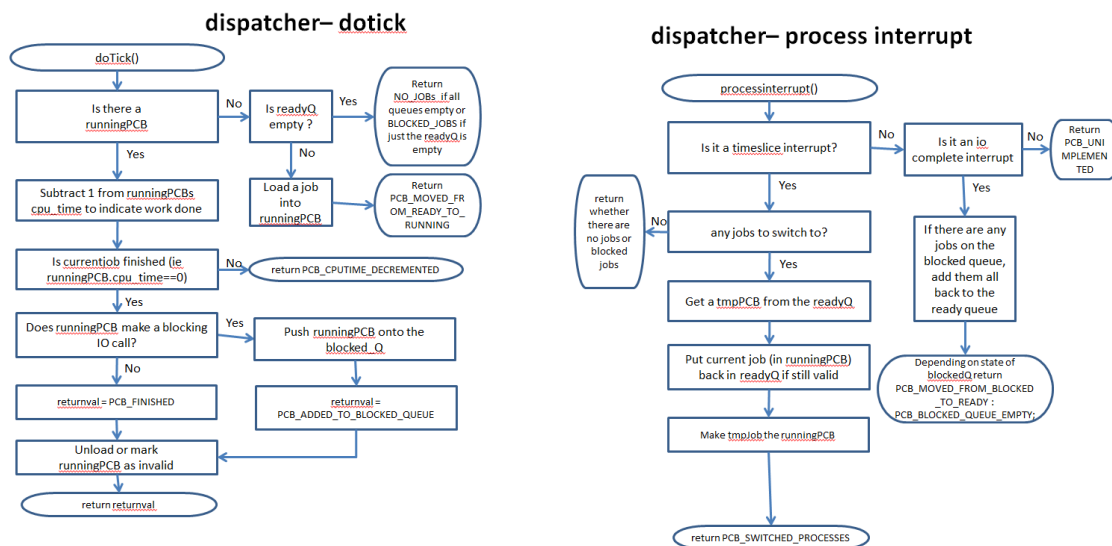
### Job List

Job list uses File\_IO to load the list of jobs to execute. Its purpose is to feed jobs to the dispatcher as they become available. You are given the header file.

Please provide an implementation in the joblist folder

## Dispatcher

The algorithmic heart. These flowcharts illustrates how its main functions should operate.



runningPCB.io\_time can be 0 or 1 as defined in the testdata.txt file. If io\_time==0 then the process has no io operations (fat chance in the real world). If io\_time ==1 then the process will have one IO operation that requires it to be moved to the blocked queue. This occurs after the cpu\_time has all been used. In other words, the cpu\_time is decremented each tick count until it reaches 0, then if io\_time is equal to 1, runningPCB is moved to the blocked queue (first set io\_time=0) where it awaits an IO\_interrupt. Once the interrupt happens it is moved back to the ready queue where it exits.

Please provide an implementation in the dispatcher folder

## Output

The program prints a lot of info to the console to help you see what is happening.

The program also logs what process is executing during which tickcount, see ST\_LOG at the end of Proj2\_Main.cpp. I will use this output to gauge your programs accuracy. I will also use different test data than what I provided.

## Documentation and Testing

Make sure you comment each function and the program as a whole. I recommend you test each module separately (which implies you write test code to verify its behavior).

Joblist should be relatively easy, it must load a file and sort the contents by start time, then when it's time to load a job it does so.

The dispatcher is harder: I recommend you start with simple data

1 process no IO

1 process with IO

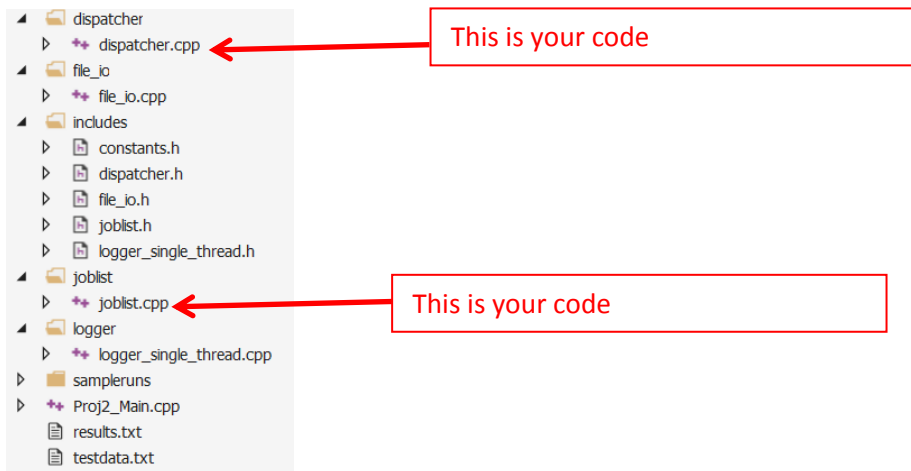
2 processes no IO  
2 processes with IO  
Then many processes with a mix of IO and nonIO processes

To help you I have included several files with sample inputs and outputs, see sampleruns folder.

All were run with these defaults as defined in Proj3\_SOL.cpp.

```
const int    TIMESLICE = 4;  
const int    NUMBER_CYCLES_BETWEEN_IO_INTERRUPTS = 20;
```

## Directory Structure



## Grading

I will drop your dispatcher.cpp and joblist.cpp into the appropriate places in my project. Therefore these are the only files that I need from you.

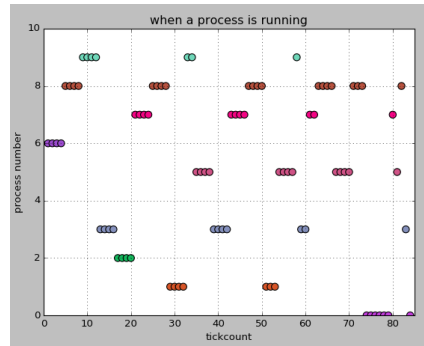
30% joblist.cpp works correctly  
30% dispatcher correct without io blocking  
40% dispatcher with io blocking

I use the output provided by the ST\_LOG call in Proj2\_Main.cpp to gauge your programs accuracy. I will also use different test data than what I provided as part of the project.

**This assignment is complex and is weighted 1.5 times the weight of project 1**

## Do you want to display your data?

I've included a python program in the plotProcesses folder. Here is what results.txt plots to using this program. It helps to visualize whats going on with your program. Use it if you want.



## APIs that may help

std::queue example

```
#include <queue>
```

```
std::queue<PCB> ready_Q;
```

```
//to add
```

```
ready_Q.push(myPCB);
```

```
//to remove
```

```
runningPCB = ready_Q.front();
```

```
ready_Q.pop();
```

```
//get its value
```

```
//remove from queue
```