SOFE 3950U / CSCI 3020U:

Operating Systems

Lab #4: Host Dispatcher Shell Project

Ramzi Haddad - 100665475

Shomari Simpson – 100581869

Abdi Ibrahim – 100582257

Design Document

Contents

[Queue.c 2](#_Toc509502313)

[Push 2](#_Toc509502314)

[Load\_dispatch 2](#_Toc509502315)

[Hostd.c 3](#_Toc509502316)

[Initializequeues 3](#_Toc509502317)

[print 3](#_Toc509502318)

[Utility.c 4](#_Toc509502319)

[Alloc\_mem 4](#_Toc509502320)

[Free\_mem 4](#_Toc509502321)

[MakeT 4](#_Toc509502322)

# Queue.c

This class is responsible to load the item from the distich list into the process queue.

## Push

This method will add the process into the process queue. It will assign the process to the dedicated queue based on the priority(all received from the load\_dispatch). A

// receive the parameters .

//add to the queue. Based on the parameter

## Load\_dispatch

This method will read the dispatch list then, it will assign the resourse based on their priority.

We will have 4 roots (which are pointer for the process structs), 4 conductors to loop through the list,

Using the push method

// Read file

// create process struct.

// assign read values into a process object

// Use push method to add to the process list as follow:

|  |  |
| --- | --- |
| if (priority==0){ | |
|  | push(root0,conductor0,process); |
|  | } |
|  | if (priority==1){ |
|  | push(root1,conductor1,process); |
|  | } |
|  | if (priority==2){ |
|  | push(root2,conductor2,process); |
|  | } |
|  | if (priority==3){ |
|  | push(root3,conductor3,process); |
|  |  |

# Hostd.c

This class creates the hosts for the host dispatcher using queues to hold the created processes.

## Initializequeues

This method initializes the queues using the defined zeroRuntimeProcess structures. It assigns values for runtime, priority, address, and pid variables of the zeroRuntimeProcess. It then assigns memory for the processes using malloc in a queue structure holding the processes.

// assign values (runtime, priority, address, and pid) for each of the processes using the zeroRuntimeProcess struct.

// set root queue equal to allocated memory of the size of the queue structure

// use root queue memory to hold the processes

// set root to conductor queue value and fill queue with next values

## print

This method will execute appropriate print statements depending on the value of the priority taken in.

// take in priority value

// if conductor for the process is not 0, execute while loop which prints the process variables (runtime, priority, address, and pid)

// print process variables

# Utility.c

## Alloc\_mem

This method will allocate process to a free memory, basically, it will check for a free memory , if there is a free one , it will assign the resource to it, otherwise it will delete some process and then replace it with the given process from the queue.

// check for free space

// If yes assign the task to the free space.

// If no loop the through the queue of the resources and return the address.

// do the last for all of the resources list (3 priorities 3 lists)

## Free\_mem

This method will add more space for the available resources.

// reset the resource memory to zero

// add the resource size to the available memory.

## MakeT

This method will terminate a process from the queue (4 lists).

// go through the list to the end based on the list.

// make termination.

// increase the number of terminations for that list.