### **Overall State Diagram**

### 

### **Server State Diagram**

**Client State Diagram**

**Pseudocode**

### **Process Argv**

Catch the SIGINT signal when user enters ^C, which will terminate the process and close the message queue

Argv will contain the type of service, file name, and priority level

Validate the command line arguments

If user entered “-s”, valid

Transition into **Get Request form queue (Server)** state

If user entered “-c” and number of arguments is equal to 4, valid

Transition into **Query for Connection (Client)** state

### **Server:**

### **Get request from queue**

Forever loop:

Reads on blocking for a new client query from the message queue

-mtype specified as a global defined variable

-the query message will contain the client’s process ID

When the query is received

Transition to **Respond with PID** state

### **Respond with PID**

Initialize the message structure

-mtype specified as a global defined variable

-message content will contain the server’s process ID

Open existing message queue

Write the message structure to the queue

Create a child process to handle the new client’s communication

Transition to **Child Process** state

When child process ends, make sure to break and terminate it

### **Child Process**

Initialize a message structure

Read from message queue

-mtype specified as the client’s process ID

-message content will contain the file name and its priority level

When the message is read

Transition to **Open & Read File** state

### **Open & Read File**

Attempt to open the file name passed from the message structure

If the file does not exist

Write a NULL message back onto the queue

-mtype specified as the client’s process ID

-message content will contain NULL

-exist child process

Start reading the file content

Have a dynamically sized string that will store the file’s content

Transition to **Send File to Queue** state

### **Send file to Queue**

Calculate the size of content that will be sent to the client

-size is based off of the priority level, higher priority = bigger content

Loop through the buffer until there are no data to be sent

Initialize the message structure that will be written to the client

-mtype specified as the client’s process ID

-message content will contain the sliced chunk of data based off of the size calculated

Send the message structure to the client

Transition to **END,** exists child process

### **Client:**

### **Query for connection**

Initialize message structure to communicate with the server

-mtype specified as a global defined variable

-message content will contain client’s process ID

Send structure to message queue

Transition to **Get Response from queue** state

### **Get response from queue**

Initialize message structure for reading from the queue

Reads from the message queue

-mtype specified as a global defined variable

-message content will contain server’s process ID

If message content contains a null character

Transition to **Remove Queue** state

Transition to **Send file name + priority** state

### **Send file name + priority**

Sends message structure to message queue

-mtype specified as the client’s process ID

-message content contains the file name to open and its priority level

Transition to **Read from queue** state

### **Read from queue**

Read message structure from message queue

-mtype contains the client’s process ID

-message content contains the the file content block

Transition to **Display content** state

### **Display content**

Loop:

Keep displaying content sent from the **Read from queue** state

If the size of the content block is smaller than the size calculated based off of the priority level

Transition to **Remove queue** state

Transition back to **Read from queue** state

### **Remove queue**

Find the existing queue and close it

Exists the application