Resource Manager Documentation

# Program Design

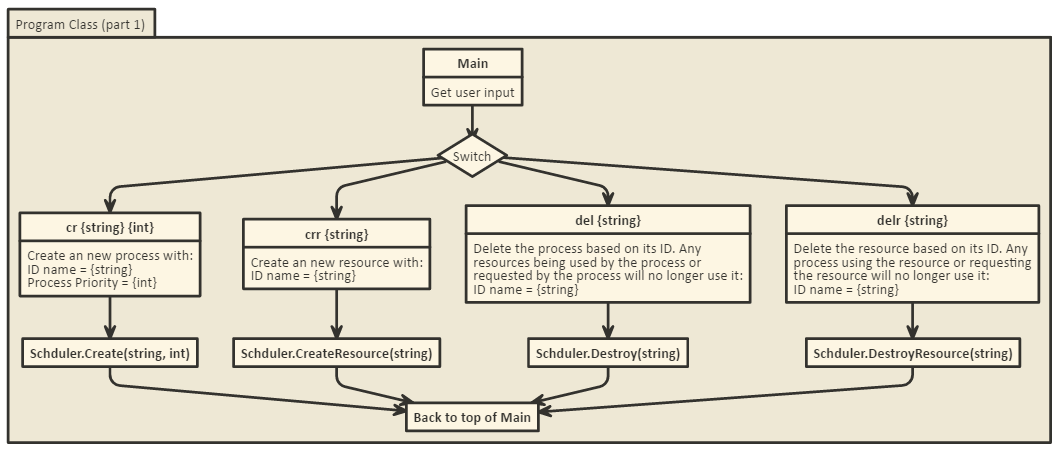
This program simulates the user creating process and resources as well as the allocation of those resources. This simulation uses a scheduler class that goes through the process list after every request is down and sees if the current process running should be which based on its current blocking status and its priority.

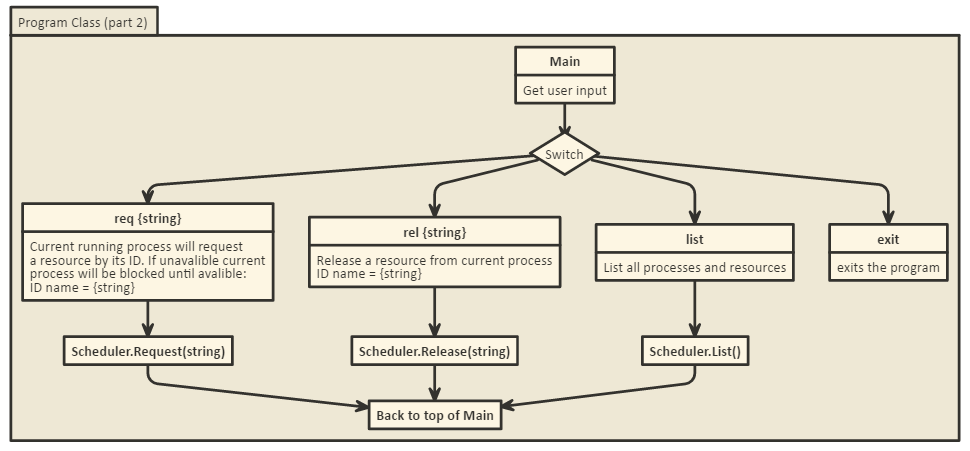
This is the basic logic of the program:

* Get the users input
* Parse input to execute a command
* Run the command (Some kind of create, delete or modify statement usually)
* Reanalyze the current process list and run the highest priority that has the resources needed available.
* Repeat.

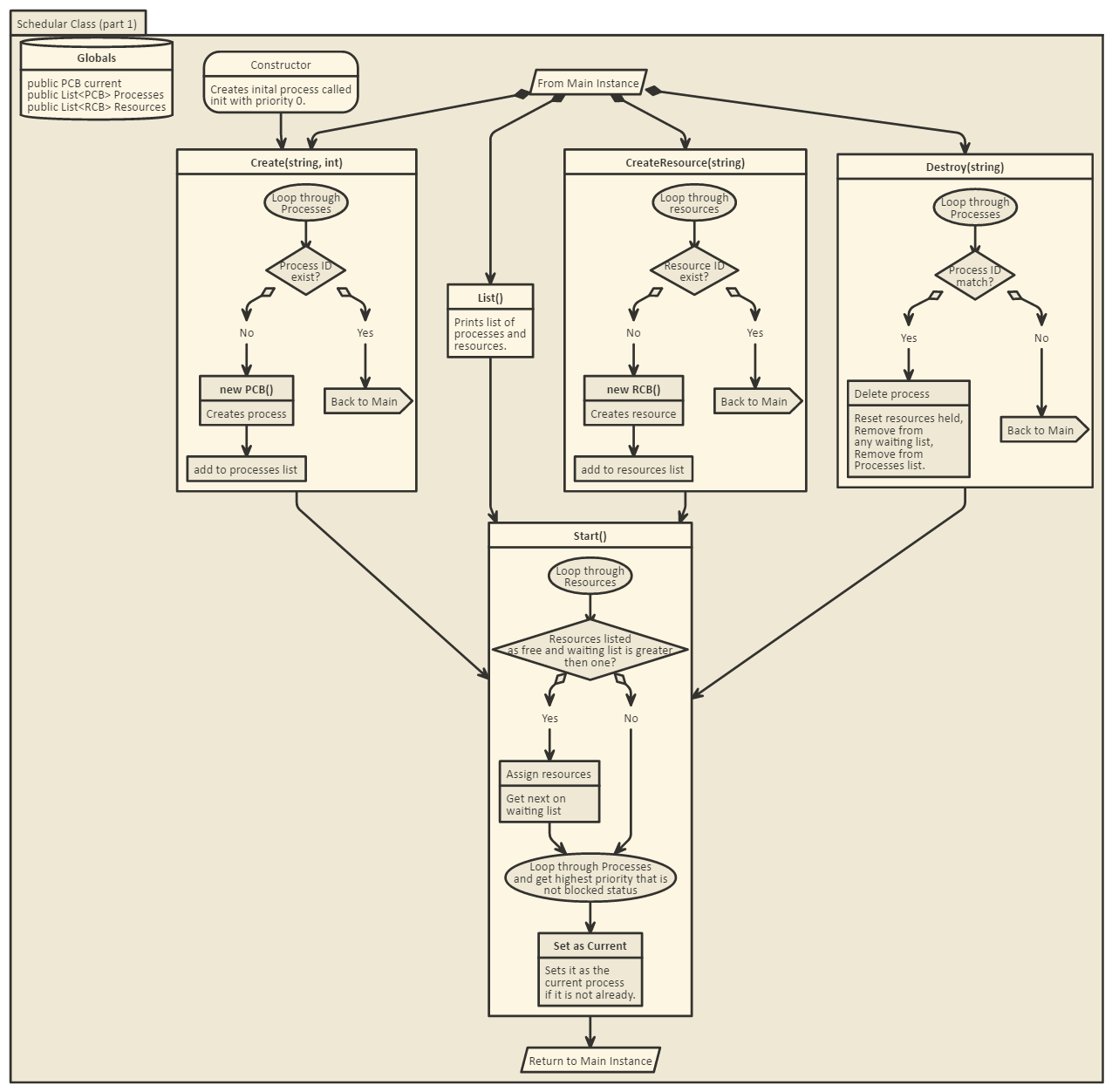
The overall logic of the program is simple really; the real meat of this application is in the modification functions and the control block structures. Adding new functions to this system would not be too difficult but based on the functionality may require some modification to some of the other functions in the scheduler class.

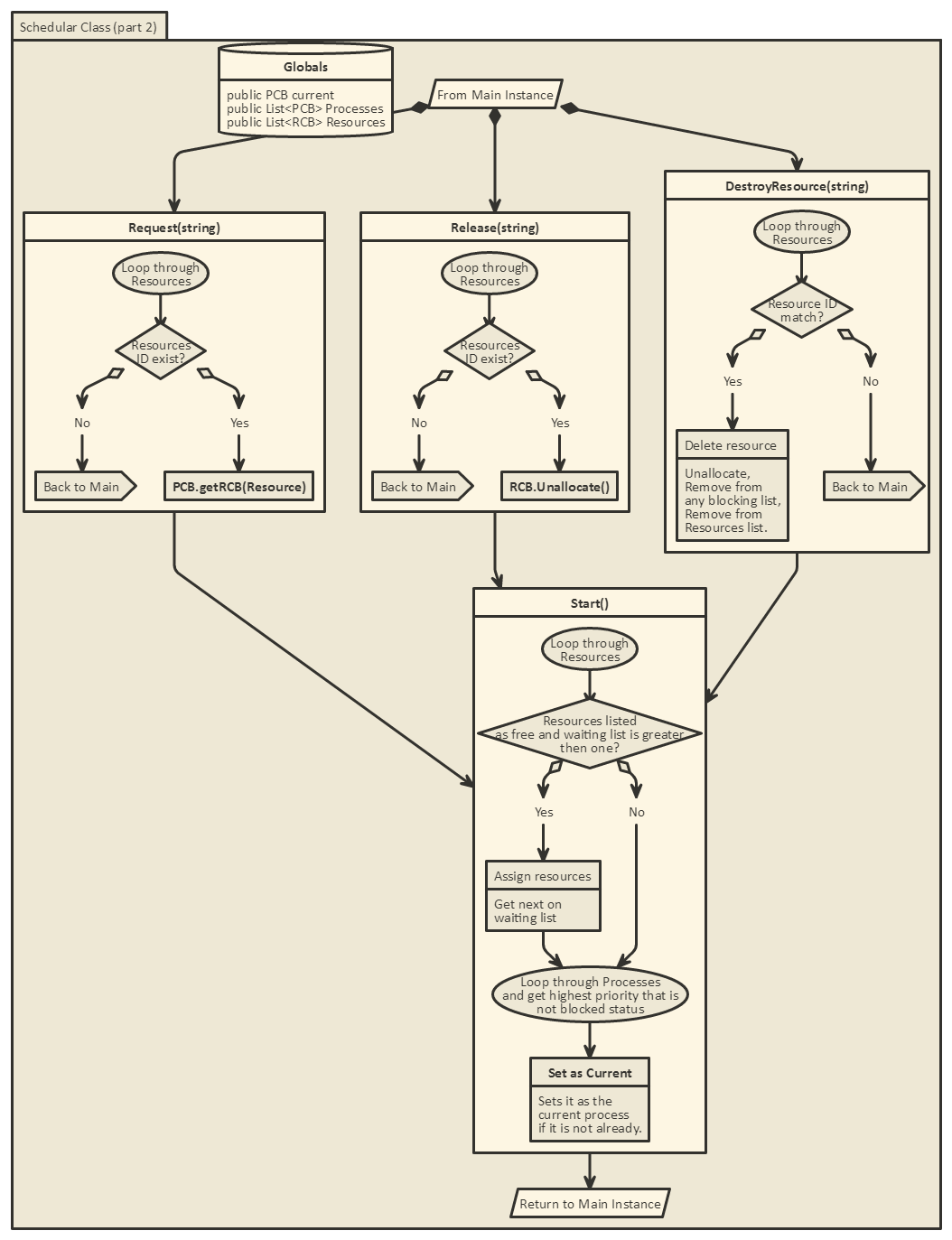
The following are diagrams of every class:



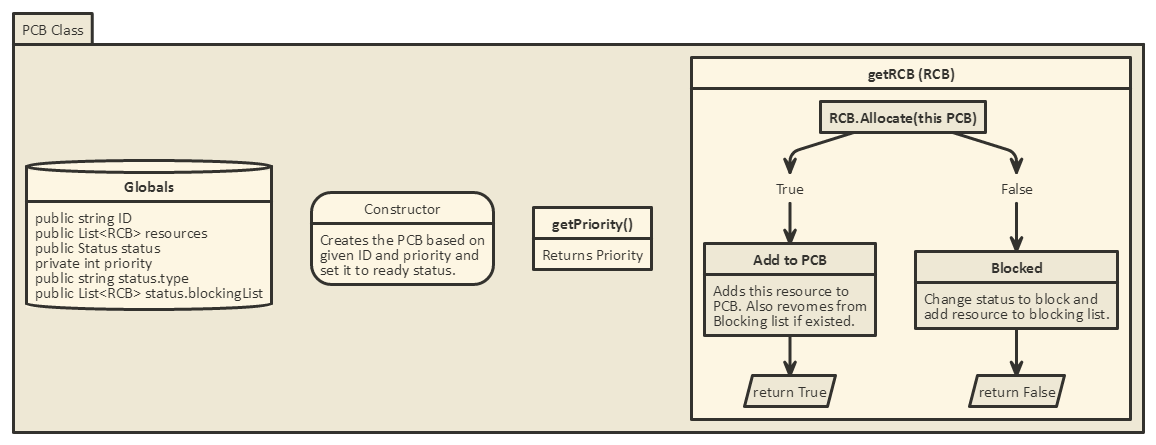


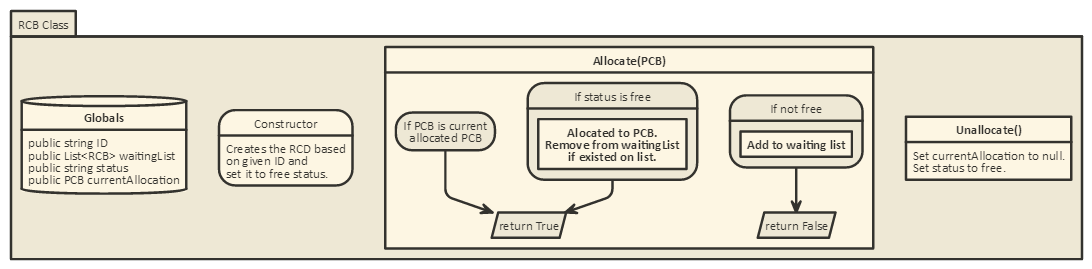
The main Program class just handles all of the inputs and parsing of strings. It dates the entered data into a switch case and then executes the appropriate scheduler function. The while loop that it goes in will only exit with the ‘exit’ command entered as an input.





The scheduler class is where all of the operations and process/resource modifications happen. Each function is called from the main from the switch statement. They procced to perform the operation and then call the start function which will reevaluate free resources and see if any process request those resources. It will also evaluation the priority tree and run the function with the highest priority and all resources available.

The PCB class is the object for the processes. At creation they are given an ID and a priority. The only function here that really makes any modifications to the object is the getRCB() which checks and sees if the resource is available. If it is it will get added to the resource list, if not then it goes on the blocking list.



The RCB class is the object for the resources. They are defined with an ID at creation. Allocate() simple sees if the resource is free and allocates it if is to the requested PCB. If not the PCB goes on the waiting list. Unallocated just drops the allocated PCB and sets the status to free.

# Files/Database Section

There are no files needed to test this program other than the exe file to run the application.

# Compile/Execution Section

This program was created in C# using .net framework 4.6. You will need the .net framework installed on your system to run it. Navigating to the bin/debug will give you the .exe needed to run the program. Also the entire solution can be open in visual studio.

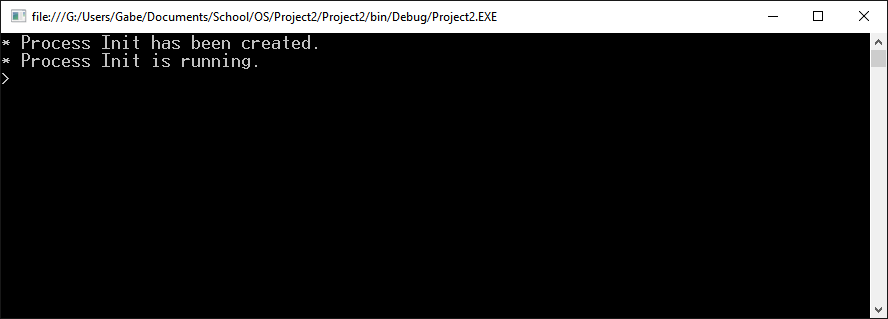
The following is a table with all the functions a user can enter into the program:  
(Note: the parameters are entered in without the (). Please only include one space between parameters and/or function name.)

|  |  |
| --- | --- |
| Function (parameter 1) (parameter 2) |  |
| cr (ID) (prior) | Creates a process with ID = (ID) and a priority = (prior) |
| crr (ID) | Creates a resource with ID = (ID) |
| del (ID) | Deletes the process with ID = (ID) |
| delr (ID) | Deletes the resource with ID = (ID) |
| req (ID) | Requests the resource with ID = (ID) to the current process |
| rel (ID) | Releases the resource with ID = (ID) from any processes |
| list | List all the process and resources with their details |
| exit | exits the application |

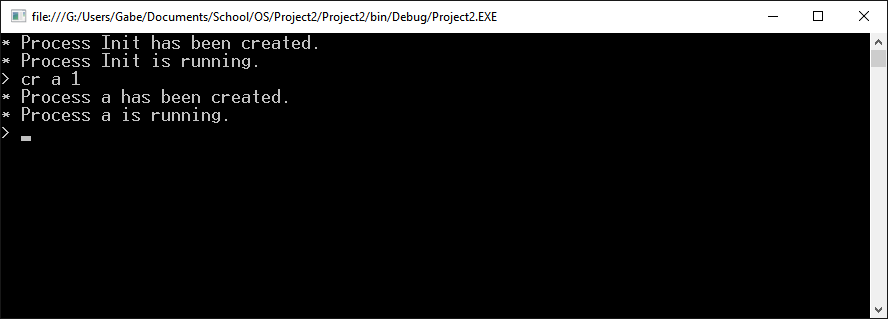
# Test Cases Section

As far as preventing invalid errors go, the program will just ignore anything it parses from the user input that is not a valid command.

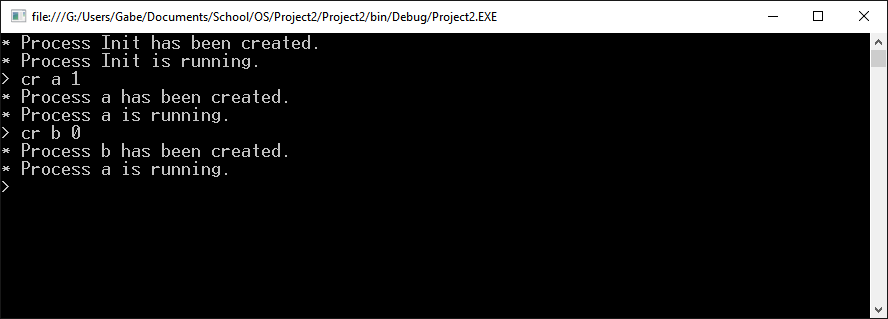
Now the following is a demonstration of all the operations working as intend. First Lets start the application.



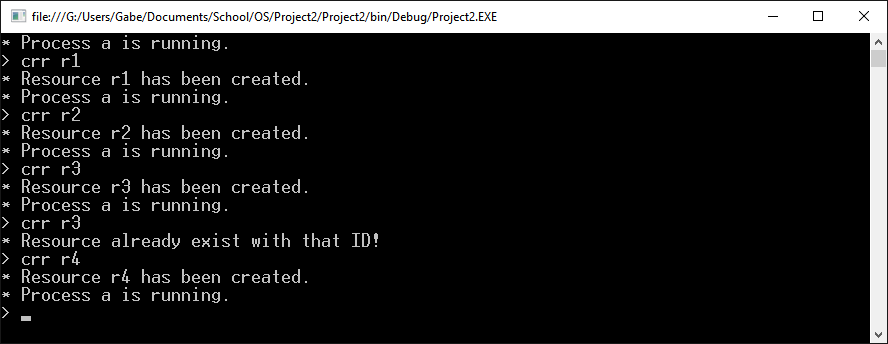
Here we can see that an initial process being generated this is running at priority zero. Let’s create a new process with priority 1 called “a”.



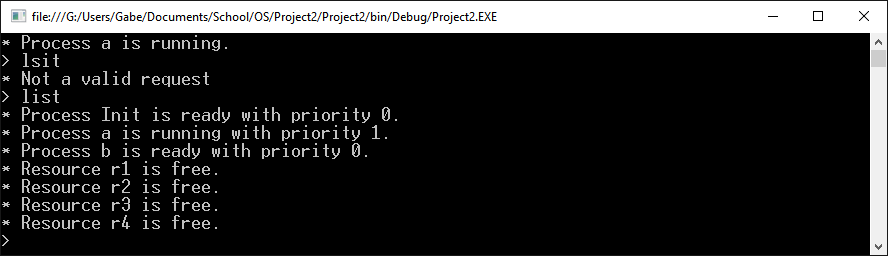
Here the process “a” is created and since it has a higher priority than the init process it will run instead. Now let’s try a lower priority process.



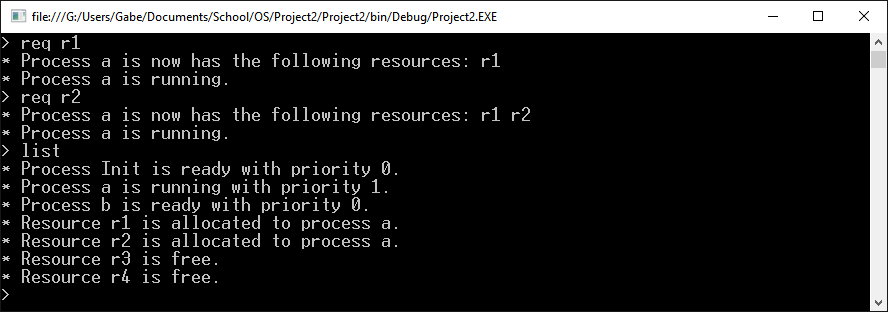
Here process “b” is created. But since its priority is 0 and lower then a’s then it will not run until a no longer exists. Next lets make 4 resources with the “crr” command.



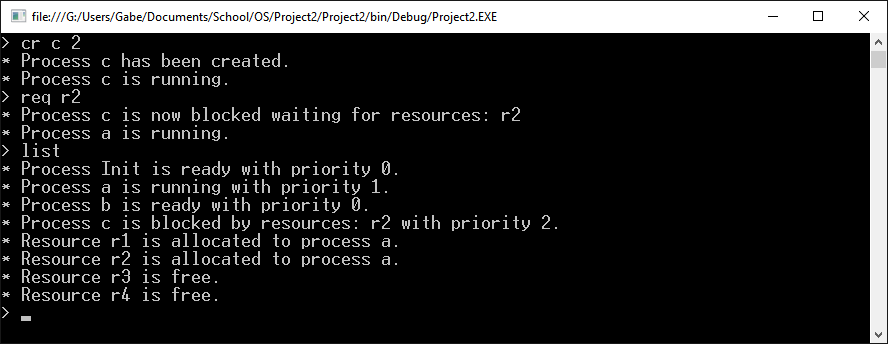
Here we can see that I was able to create 4 new resources. I also tried to create r3 twice, but the program told me that it was already existed and did not create it. Now with all those resources and process create let’s look at them with the “list” function.



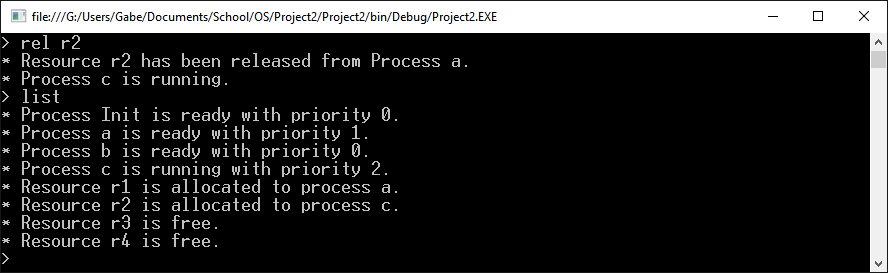
First I entered a typo and the program said that it was not a valid command. Then I print the list of process and resources. Now we can see what to allocate. Let’s allocate r1 and r2 to the current process and reprint the list.



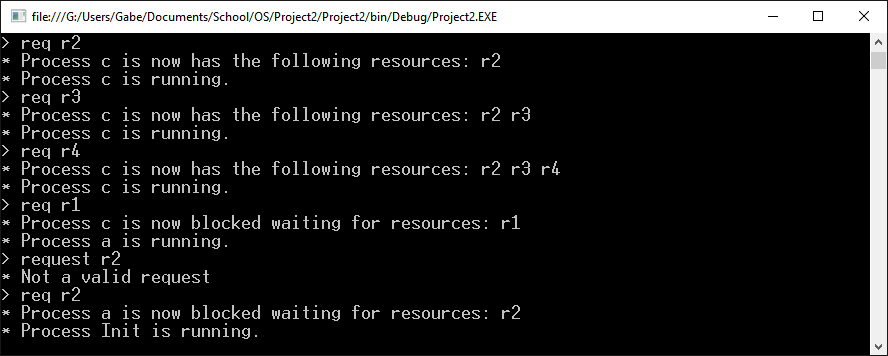
Now we see process a having resources r1 and r2. Next lets create a new process and allocate r2 to it.



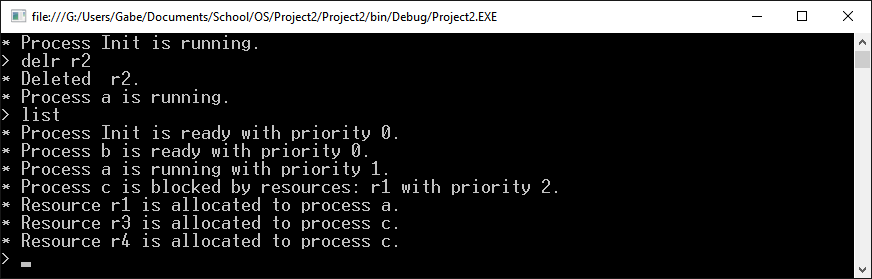
We can see that process c started running with the priority of 2 but then after requesting r2 was block as a has r2, so a started running again. Now let’s release r2 so c can use it.



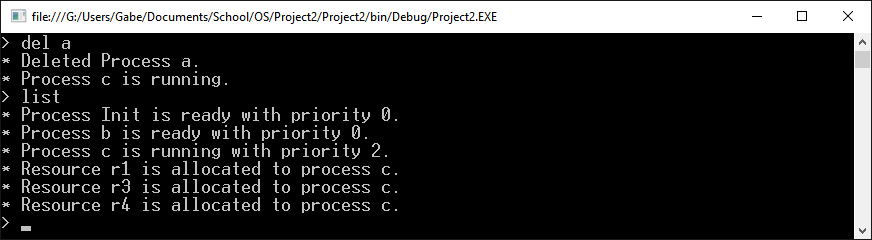
Here c was able to start running again and was able to grab r2. Now lets grab some more resources for c and some more for a.



Now we can see that both a and c are block so process Init will run since it was the first created with priority 0. Now let’s release a resource by removing it completely with “delr”.



Since r2 was deleted no process will try and hold it as a resource anymore and a could run. No the last way to release a resource request to another process is to delete the process holding it. So let’s delete a with “del”.



So now that A is removed, resource r1 is free and can be grabbed by process c. C now has all of its resources and is the highest priority so it runs next.

That is the basic use case for this application and it does run as intended. The resources release is fairly robust and I have only found an issue on potentially locking all processes so none can hold the currently running status. I override this by just defaulting the current process to the first created. Just so the user can create another process. As soon as that other process is created it will jump to that instead.