OS Lab 4, Section 1: Formal Design

Faculty of Engineering and Applied Science

SOFE3950U: Operating Systems| CRN: 74025| Section: 001

Due: March 15th, 2024

Group 31

Logan Butler (100828103)

Ontario Tech University

Oshawa, Ontario

logan.butler@ontariotechu.net

Dmitri Rios Nadeau (100783206)

Ontario Tech University

Oshawa, Ontario

ericdmitri.riosnadeau@ontariotechu.net

Nathan Perez (100754066)

Ontario Tech University

Oshawa, Ontario

nathan.perez@ontariotechu.net

Jordan Frost Hagedorn(100828122)

Ontario Tech University

Oshawa, Ontario

jordan.hagedorn@ontariotechu.net

Section 1:

Describe and discuss what memory allocation algorithms you could have used and justify your final design choice.

For memory management, the code dynamically allocates memory using the malloc command in combination with the size of the "Job". Jobs are the structures hold all of the information needed for processing are found withing that structure. To deallocate the memory, free() is called to ensure that no memory leaks. Other methods of memory allocation include first fit, where the memory is separated into blocks and the first block that can fit this job, is where it will be stored.

Describe and discuss the structures used by the dispatcher for queuing, dispatching and allocating memory and other resources.

The entire structure is modeled by seven linked lists. The fist one represents the dispatch list and is generated in order by the txt file provided by the user. The second and third one represents the real time queue and the user job queue. The fourth fifth and sixth linked lists represent the the priority one two and three queues. The last linked list represents the processor. For allocating memory we used global variables and a new struct that kept track of currently used resources. To check if the system could fit the processes onto the "ready to process" queues, the system would check if the process would overload the system, using the total resources in combination with the resources already being used. If the job can fit in the system, its resources are added to the pool of used resources, and when the task finishes, the resources are subtracted from the pool of available resources.

Describe and justify the overall structure of your program, describing the various modules and major functions.

Essentially the main guts of the program happen in one while loop. This while loop will run until the max processes time has been reached (which is currently set to 20, and found in the MEManage.h header file). Every tick of the loop, if there is a current process running, it will reduce its remaining process time (Free resources if task is finished) and increase its priority level (if its not zero and below 3, and if it has reached its designated quantum time). The program then checks if there are any processes with priority level integer lower than itself, and if there is, the current running process is added to the appropriate queue and the new process takes over on the processor. At the end of every tick, the queues are printed out to show whats in them.

Discuss why such a multilevel dispatching scheme would be used, comparing it with schemes used by "real" operating systems. Outline shortcomings in such a scheme, suggesting possible improvements. Include the memory and resource allocation schemes in your discussions.

In the system, for all of the priority queues, they have the similar time quantum. This can lead to priority one and two jobs getting lost in the sea of unfinished processes left in priority three. To fix this, the time quantums for the priority one and two should be significantly greater than the time quantum for priority three. This could also lead to higher level processes not being able to come in after priority 3 queue steals all of the resources such as memory and IO devices.