**Technical ReadMe**

**CPU Scheduler**

Version 1.0

1. **Program Goals**

The goal of this program is to simulate the implemented CPU Schedulers that is available in this program. With each algorithm, they carry their own unique protocols such as organizing shortest CPU Burst time (*Shortest Job First*), organizing critical processes first (*Priority*), execute the processes as they arrive (*First Come, First Serve*), and execute processes by CPU quantum leveling (*Round Robin*).

This program can support up-to 9000 processes and a minimum of 100 processes, which the user can experiment with the results of each algorithm.

The results will be presented in two forms:

* Terminal Buffer
* Log file

This allows the user to inspect and analysis the results provided by this program.

1. **Requirements**

This program is capable of being compiled across platforms. With that in mind, the end-user can run this program in either GNU/Linux or Microsoft Windows, provided that - they compile the program in the desired platform. The libraries used in this program are basic to the C language and API’s, but not entirely dedicated to one specific platform.

Additionally, this program may require at least 10 megabytes of RAM. At some testing’s, I have noticed that I was taking at least six – eight megabytes of RAM during operation. Though, I don’t fully remember if that was before or after I set the limit of maximum processes possible.

1. **Debug Mode**

This program supports various debug modes. The debug modes available in the program cannot be toggled during runtime, instead – they are hardcoded by using macro’s in the source code. The end-user is fully allowed to alter the source code as they see fit, and this includes the debug mode switches.

**Debug Mode Switches**

|  |  |  |  |
| --- | --- | --- | --- |
| **LINE #** | **Macro Keyword** | **Default Value** | **Description** |
| 131 | \_DEBUG\_PROCCHAIN\_ | False | When true, this will show each individual process properties within the Process Chain. However, when true, this can degrade performance during runtime. |
| 134 | \_DEBUG\_CALC\_ | False | When true, this will show the Average Waiting Time and Average Turn Around Time for each pass when Process Chain has been generated. This should not be confused with the Overall Calculation. |
| 135 | \_DEBUG\_STEPWISE\_ | False | When true, this will tell the user which step the program is currently performing or the operation that is being executed at real time. For example, if the Shortest Job First operation is being executed, then a message will be displayed on the screen that it is presently being executed. |

**Debug Mode Demonstrations**

|  |  |
| --- | --- |
| **Debug Macro** | **Demonstration** |
| \_DEBUG\_PROCCHAIN\_ |  |
| \_DEBUG\_CALC\_ |  |
| \_DEBUG\_STEPWISE\_ |  |

1. **Return Codes**

This program relies on necessary components to be active and ready for usage. When one component is not working correctly, the program could potentially terminate with an error. Listed below is the exit codes that this program will return to the host Operating System:

|  |  |
| --- | --- |
| **Return Code** | **Explanation** |
| 0 | Clean exit; no errors |
| 1 | Unable to open or create log file |
| 2 | Failed to allocate memory to heap |

1. **Logging**

This program does support auto-logging, depending if the macro switch \_OUTPUT\_FILE\_FLAG\_ is set to ‘True’. When logging is enabled, all critical messages that is displayed on the screen will also be logged in a log file. However, it is recommended to enable one or more debug switches when logging. All Debug modes that is implemented in this program will not only display on the screen but will also be logged in a text file. If all Debug switches are disabled, logging will be extremely minimal – adjacent to what is being displayed on the screen at real time.

*Worst File Size Risk: ~300MB*

*Recommended HDD Size: 500MB*

1. **Sample Use**





