Power Consumption Monitor

A Node.js command-line application that allows users to track a specific process and shows its CPU power consumption based on CPU usage.

- Language used: *Javascript*
- Runtime: *NodeJS*
- Libraries used: *os, readline, process, child_process, google-it*

Installation

To install the dependencies run the following command:

```
npm install
```

Usage

To start the application, run the following command:

```
node index.js
```

Once the application starts, you will see a list of processes currently running on your machine. Use the `arrow keys` to navigate the list and press `enter` to select a process.

You can also search for a specific process by typing in the `name` or `pid` of the process.

Documentation

Functions

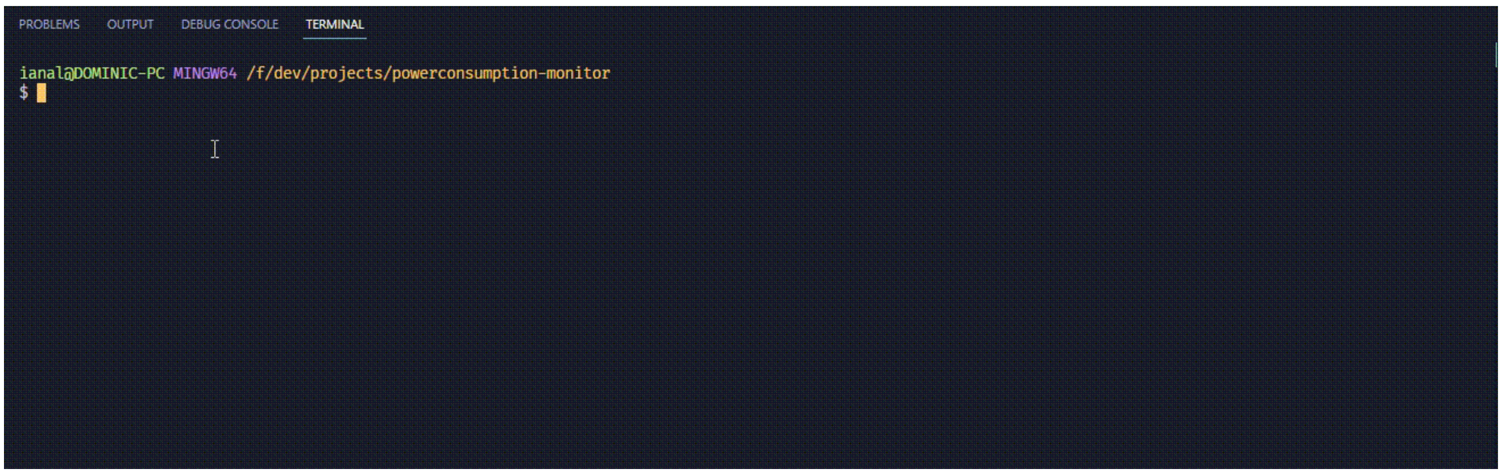
- `main`
  - **Parameters**: none
  - **Return**: none
  - **Description**: This is the main function of the program. It is where the program starts and where the user is prompted to select a process to track.
- `getProcessList`
  - **Parameters**: none
  - **Return**: Promise that resolves to an array of strings containing the names of all currently running processes on the system.
  - **Description**: Uses the `exec` method of the `child_process` module to run the `tasklist` command and retrieve the list of running processes.
- `printList`
  - **Parameters**:
    - `list`: an array of strings containing the items to be printed.
    - `selectedIndex`: an integer representing the index of the currently selected item.
    - `search` *(optional)*: a string representing the current search query.
    - `itemsToShow` *(optional)*: an integer representing the maximum number of items to be displayed.
  - **Return**: none
  - **Description**: Prints the list of items in a formatted way on the console. The selected item is highlighted with a '>' symbol.
- `getInfo`
  - **Parameters**:
    - `selectedProcess`: a string representing the details of the process whose information is to be retrieved.
  - **Return**: Promise that resolves to an array containing the `name` and `PID` of the specified process.
  - **Description**: Converts the string into an array of words and retrieves the `name` and `PID` of the process.
- `getProcessCpuUsage`
  - **Parameters**:
    - `pid`: an integer representing the PID of the process to monitor.
  - **Return**: Promise that resolves to a number representing the percentage of CPU usage by the specified process.
  - **Description**: Uses the `exec` method of the `child_process` module to run the `wmic` command to retrieve information and compute the CPU usage of the specified process.

- `estimateProcessPowerUsage`
  - **Parameters**:
    - `cpuUsage`: a number representing the percentage of CPU usage by the process.
    - `TDP`: the *thermal design power*, in watts, of the CPU. It refers to the power consumption of the CPU under the maximum theoretical load.
  - **Return**: a number representing the estimated power usage of the process in Watts.
  - **Description**: Calculates the estimated power usage of the process using a formula that takes into account the CPU usage and the TDP of the CPU.
- `trackProcess`
  - **Parameters**:
    - `processName`: a string representing the name of the process to monitor.
    - `processId`: an integer representing the PID of the process to monitor.
  - **Return**: none
  - **Description**: Continuously monitors the CPU usage of the specified process and estimates its power usage. Prints the estimated power usage on the console.
- `searchTDP`
  - **Parameters**:
    - `cpuModel`: a string representing the model of the CPU.
  - **Return**: Promise that resolves to a number representing the TDP (Thermal Design Power) of the CPU.
  - **Description**: Searches for the TDP of the specified CPU model on Google using the google module.

Program flow

1. The program starts by calling the `main` function, which clears the console, gets the list of running processes using the `getProcessList` function, initializes variables, and prints the list using the `printList` function.
2. The program listens for user input using the readline module. The user can navigate through the list of running processes using the `arrow` keys and select a process to track by pressing the `Enter` key. The user can also `search` processes by typing the query.
3. Once a process is selected, the `getInfo` function is called, which retrieves the process name and process ID of the selected process.
4. The `trackProcess` function is then called, which continuously monitors the selected process using the `getProcessCpuUsage` and `estimateProcessPowerUsage` functions.
5. The `getProcessCpuUsage` function retrieves the current CPU usage of the selected process.
6. The `estimateProcessPowerUsage` function estimates the power usage of the selected process using the TDP (Thermal Design Power) of the CPU, the current CPU usage of the selected process, and the total number of CPU cores.
7. The `searchTDP` function searches for the TDP of the CPU model using the Google search API.
8. The estimated power usage of the selected process is then printed to the console.
9. The `trackProcess` function repeats steps 5 to 8 until the user manually stops the program by pressing `Ctrl+C`.
10. The program exits.

Demo



```
ianal@DOMINIC-PC MINGW64 /f/dev/projects/powerconsumption-monitor
$ 
```