main.c

Double absval(double num)

- Takes in a double num and returns the absolute value of num

Int isNumeric(char *str)

- Given a string, determines if the string is a number
- Returns 1 if can be converted to a number, 0 otherwise

Int getNumericValue(char *str)

- Given a valid string, returns the numeric value of the “number”

Float kiloToGiga(long kilobytes)

- Takes in a long int of kilobytes and returns the conversion to gigabyte with the formula gigabyte = kilobyte/(1000^3)

Int main(int arg c, char *argv[]) **Main Function**

- Takes in arguments and argument count
- Loops through provided arguments in terminal, if argument is valid, eg. “--user”, “--system”, “--sequential”, “--samples=N” where N is a valid number, “--tdelay=T” where T is a valid number, then it will following instructions accordingly, and “--graphics” or “-g”
- If “--user” is present, variable user is set to 1 (default user = 0)
- If “--system” is present, variable systems is set to 1 (default systems = 0)
- If “--sequential” is present, variable user is set to 1 (default sequential = 0)
- If ints user == systems, then program treats as if those flags were never used
- If user == 1 && systems == 0, only user statistics will be outputted, and vice versa
- If “--samples=” and “--tdelay” both not specified, the first two arguments will be positioned for N samples and T tdelays respectively
- If int sequential == 1, then output will be done sequentially and not refreshed on one screen
- All outputs would done with N samples every T seconds, if not specified, by default N=10 & T=1
- If “--graphics” or “-g” is present, int graphics gets set to 1 (default graphics = 0)
- If graphics == 1, then system statistics will be outputted with graphical representations

Memory usage: `::::::@ total relative negative change`
`######* total relative positive change`

CPU usage: `||||  positive percentage increase`

cpuStats.c

```
Struct cpu{
  unsigned long user;
  unsigned long nice;
  unsigned long system;
  unsigned long idle;
  unsigned long iowait;
  unsigned long irq;
```

```
    unsigned long softirq;
}
```

All attributes of struct cpu represents the components of cpu usage

Void store_info_to_struct(cpu *pro)

- This function takes in a pointer to a cpu struct and stores the values of each component after reading from the “/proc/stat” file

double get_total_usage(cpu *cpu)

- Given a pointer to a cpu struct, it computes and returns the total cpu capacity by summing all components

double get_cpu_usage(cpu *prev, cpu *cur)

- Given two pointers to cpu structs, prev represents the last sample and cur represent the current sample, this function computes and returns the total usage
- My calculation was based off of a website a TA provided at https://www.kgoettler.com/post/proc-stat/
- I think this calculation makes sense as it compares the ratio of the previous and current usage, specifically the idle and iowait usage, and the total usage, to determine usage by the formula 100 * (diff of total - diff of idle & iowait)/ (diff of total)

systemStats.c

void printSystemStats()

- Prints out the machine and system information by accessing <sys/utsname.h> and using function uname(struct *utsname)

userStats

void printUserStats()

- Reads through utmp file and prints out the user and its information in real time

How to use this program:

- To use this program, compile the main.c file and run along with any valid arguments you desire.
- If no arguments were added, the program prints out all information non-graphically by default with a sample size of 10 and frequency of every second
- If the first two arguments were positive integers (DO NOT ENTER NEGATIVE OR ZERO), then those integers would be treated as the sample size and frequency respectively
- If only the first argument is a valid integer, it will be treated as the sample size
- “--user” will only display user information
- “--system” will only display memory and cpu usage
- “--sequential” will cause the program to output new samples per iteration
- “--graphics” or “-g” will display output graphically
- “--samples=N” where N is a valid number will set the sample size to N
- “--tdelay=T” where T is a valid integer will set the frequency to 1 sample every T seconds

- The order of the flags do not matter except for the positional arguments of sample size and delay

How I solved the problem:

- To solve this assignment, I first watched the demo then read all the provided linux documentation to figure out what to use for each component of the assignment
- In addition, using Piazza I found a helpful link at https://www.kgoettler.com/post/proc-stat/ to be used to calculate cpu usage
- The program recognizes what to do as I first loop through the arguments and check if they are flagged
- Then according to the flagged arguments, I use logic to tell the program how to act with a certain sequence of commands
- I also implemented the graphics aspect to obtain bonus marks
- I also wrote a bunch of helper functions to convert and parse information with the documentation at the top of this report
- I didn't follow Marcelo's format, instead, I printed the memory and cpu usage side by side to show they are both updated at the same time in real time, just a personal style, it looks more even