Author: Samuel Brucker

SysInfo Version 2.4

4/30/24

REDOING IMAGES

PICTURES ARE PENDING

Description:

SysInfo is a small utility tool built in C. Its purpose is to provide basic system information. With the new Version 2 release, there are expanded tools included for CPU information.

Setting it up:

The program must be ran on a Debian machine. The package “bc” is required to use option 8. If your machine does not have it, it will be automatically installed by the program, just make sure you have at least 1GB of free storage and an Internet connection.

To run the script, open your terminal and navigate to the folder containing the tool. This can be done through the use of the “cd” command. Then, compile the program using gcc.

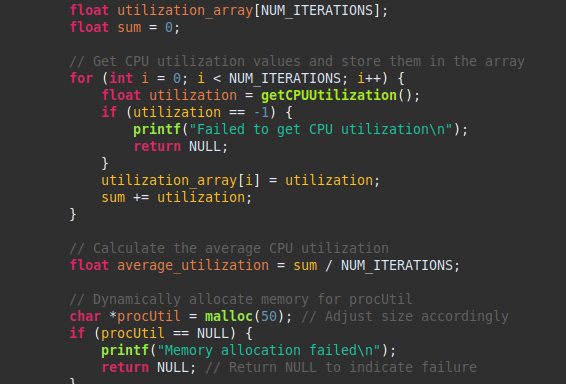
Here’s an example: “gcc main.c -o run”

If you used the example above, you can run the script by running the command “./run”, however if you didn’t use that exact command, the format to run the script is “./[name of compiled file]”. Please note this is the name of the compiled file, *not* the original main.c file.

Array:

This program uses an array for option 8. This options runs a Bash script to acquire CPU utilization ten times, storing the outputs into an array. It then provides the array, as well as the average utilization.

Here is the array’s code:



Here is the output:

Options:

There are two main sections for the options, Miscellaneous, which include options 1-7, and CPU Options, which contains 8-10. Finally, there are two options outside of either section, 11 and 12, which are respectively used to check for the requisite packages and to exit the program.

**Please see the Bash section below to see more details on how options that run bash scripts work with their respective scripts. This sections describes the operations and I/O relevant to the C file only.**

Option 1 – Time and Date:  
This option grabs the current system time and date.

Option 2 – Hostname:  
Option 2 grabs the hostname of the system, which can be thought of as an internal designator/name for the machine.

Option 3 – System Release Information:

This gives the information regarding what version the operating system is running. As this document is being written on a Kali Linux machine, this does not show up as Ubuntu.

Option 4 – Kernel Version:

Another simple option, this displays the kernel version of the system.

Option 5 – Total System Memory:  
This option displays the amount of memory (RAM) that is on the machine. This is not the amount that is available, this is the total.

Option 6 – Show Terminal History  
This option modifies a bash script, *historyOG.sh,* and creates a new, edited copy. The script is simple, showing recent lines of terminal history. The number of lines shown is specified by the user. While the C program creates the copy of the original script, it edits code to show the number of lines set by the user. This is stored as *historyNEW.sh*.

Option 7 – Confirm Script Modification  
This option is paired with Option 6 and is used to confirm that the *historyNEW.sh* script exists and is a modified version of *historyOG.sh*. The gives the user a simple way to check if the operations of Option 6 were successful outside of running the entire option again. This process will not work unless Option 6 has been ran prior, as there would be no copied script to check.

Option 8 – CPU Core Count:  
Option 8 utilizes the *coreCount.sh* script to find and output the number of CPU cores on the system. Note that this displays the number of cores, not the total number of threads.

Option 9 – Average CPU Temperature:  
This option works with *avgTemp.sh* to output the average temperature of your CPU cores. **This option is incompatible with virtual machines, as they largely do not have access to their own CPU temperature.**

Option 10 – Approximate CPU Total Utilization:  
To approximate the CPU utilization, Option 10 takes the outputs of *procUtil.sh,* organizes them into an array, and outputs that information as well as the average of them all.

Option 11 – Check for Packages  
Option 11 uses *checkPackages.sh* to confirm if the system has the bc and lm-sensors packages, which are needed for a few options. If they are not, the program automatically installs them.

Option 12 – Exit  
Very simply, this option exits the program.

**Bash Scripts:**  
This program uses three Bash scripts to perform a variety of operations. All bash scripts are stored in the *bash/*directory contained within the program’s directory. There is no need for the user to *chmod* them and set them as executables, this is handled automatically by the program.

The first is *avgTemp.sh*, which reads the /sys/class/thermal/thermal\_zone0/temp file to get the CPU’s temperature. It then converts this temperature from millidegrees to regular degrees Celsius. This script is incompatible with VMs.  
  
Next is *coreCount.sh*, which uses a mixture of lscpu and grep to grab the amount of cores on the system.  
  
There also is *procUtil.sh*. It’s purpose is to calculate the current CPU utilization of the system, but as it needs the “bc” package, it also checks for the package and installs it, if necessary. Of course, it has functionality built in to confirm there is enough storage space and and an Internet connection before attempting the installation.

This is followed by *historyOG.sh*, but a copy called *historyNEW.sh* is made upon running Option 6. While *historyOG.sh* is not intended to be ran and will error out, the *historyNEW.sh* copy displays the user’s terminal history. The number of lines set to be displayed in the script is dependent on the C program.

Finally, there is *checkPackages.sh*, which checks if the packages bc and lm-sensors are installed. If not, it confirms there is at least 1Gb of storage and that there is a network connection, and if both these conditions are met, automatically installs either or both packages if they are not already.