Fall 2018

Developing Soft and Parallel Programming Skills Using Project-Based Learning

Mad Scientists

Samuel Fekadu
Jason Moon
Pavel Beliaev
Christopher Wilson
Pranthi Cavuturu

Planning and Schedule:

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note	On Time ?
Jonghan Moon (Jason) (coordinator)	jmoon22@student.gsu.edu	Create planning and Scheduling chart	1 hr	none	10/3/18	Create a table that is easy to see who does what by when	Y
		Be recorded for the video	1 hr	Everyone attending the group meeting	10/3/18	Follow as Task 3 of Project_A2 laid out	Y
		Decide group meeting time and date and attend	2 hrs	none	10/3/18	Try to decide the best time for all of the member to meet up	Y
		Install and setup Raspberry Pi with Operating System	2 hrs	none	10/3/18	Do as soon as possible as it is the main part of this project	Y
		Task 4: Parallel Programming Basics	3 hrs	Raspberry Pi's setup status	10/3/18	Work with Pavel as we share this assignment	Y

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note	On Time ?
Pavel Beliaev	pbeliaev1@student.gsu.edu	Task4: foundation section	3 hrs	Reading material from icollege	10/3/18	Start ahead as Raspberry Pi does not require to be set up for this task	Y
		Task 4: Parallel Programming Basics	3 hrs	Raspberry Pi's setup status	10/3/18	Work with Jason we share this	Y

						assignment	
		Be recorded for the video	1 hr	Everyone attending the group meeting	10/3/18	Follow as Task 3 of Project_A2 laid out	Y
		Attend group meeting	2 hrs	None	10/3/18	Record the video as group and discuss and answer question	Y

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note	On Time ?
Pranthi Cavuturu	pcavuturu1@student.gsu.edu	Sending TA the Slack invitation	30 min	none	10/3/18	Send it to Kexin Ding: kding3@student.gsu. edu	Y
		Creating Github account and setting it up	2 hrs	none	10/3/18	Follow the instruction on Project_A2 to set it up as require	Y
		Be recorded for the video	1 hr	Everyone attending the group meeting	10/3/18	Follow as Task 3 of Project_A2 laid out	Y
		Attend group meeting	2 hrs	none	10/3/18	Record the video as group and discuss and answer question	Y

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note	On Time ?
Samuel Fekadu	sfekadu1@student.gsu.edu	Editing Video	2 hr	Finish recording the video of every member	10/3/18	Make it simple but easy to view. Just like Project A1 video	Y
		Recording group presentation video	1 hr	Everyone attending the group meeting	10/3/18	Follow as Task 3 of Project_A2 laid out. Also in charge of overall video production of the group	Y
		Attending group meeting	2 hr	none	10/3/18	Record the video as group and discuss and answer question	Y

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Note	On Time ?
Christopher Wilson	cwilson94@student.gsu.edu	Write report following the format	3 hr	All works are done	10/3/18	Start writing according to the format as the answers and information become available. Check once more before submitting to the coordinator	Y

		Be recorded for the video	1 hr	Everyone attending the group meeting	10/3/18	Follow as Task 3 of Project_A2 laid out	Y
		Attending group meeting	2 hr	none	10/3/18	Record the video as group and discuss and answer question	Y

Parallel Programing Skills

(5p) Identifying the components on the raspberry PI B+

1. Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
2. 1GB LPDDR2 SDRAM
3. 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
4. Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
5. Extended 40-pin GPIO header
6. Full-size HDMI
7. 4 USB 2.0 ports
8. CSI camera port for connecting a Raspberry Pi camera
9. DSI display port for connecting a Raspberry Pi touchscreen display
10. 4-pole stereo output and composite video port
11. Micro SD port for loading your operating system and storing data
12. 5V/2.5A DC power input
13. Power-over-Ethernet (PoE) support (requires separate PoE HAT)

(5p) How many cores does the Raspberry Pi's B+ CPU have?

4 cores

(8p) List four main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify you answer and use your own words (do not copy and past)

- ARM and X86 have different processing powers. ARM processors aim for simple instructions. However, with simple instructions, more are required to do tasks. This results in the use of more memory space and execution takes longer. X86 processors have complex instructions with more flexibility.
- There are differences in power consumption. ARM cores do not require heat sinks, so they have very low power consumption. There are fewer transistors and relatively lower speeds. Intel cores consume a lot more power because of their complexity.
- There is a difference in software availability. ARM devices can run operating systems designed for phones. X86 based devices can run any operating system that can run a PC. Both can run same applications if a virtual machine is used. Some Linux distributions exist for ARM but most operating systems are written for Intel.
- Last but not least, cost. With more complexity of X86 compared to the ARM processor, X86 and X86-64 based processors are more expensive. And with the difference in the cost, the primary purpose for the processors differ as well. Intel's X86 based processors are generally used for heavy to medium tasks, such as in research fields, server management field, and even home computer. However, ARM processors are more portable and affordable and are made for devices in which daily tasks are medium to low performance (such as smartphones and low-end computer such as Raspberry Pi, chromebooks, tablets, and other low-performing devices).

(6p) What is the difference between sequential and parallel and identify the practical significance of each.

A broken down series of instructions are executed one after the other in sequential computation. Only one instruction may execute at any moment in time. The practical significance of sequential programing is in its processing order, sequential or serial processing is very important when you want your program to run in uniformed order. For example, the reason many modern video games don't utilize parallel programming is due to risk of desyncing issue in between each frames. The will cause anomaly such as screen tearing. .Parallel computation is the use of multiple resources to solve a computational problem. The broken down sets of instructions are solved at the same time on different processors. The practical significance of parallel programming lies with processing intense works, such as 3-D modelling, video rendering, and server managing. Mentioned tasks above are heavily CPU dependent. Those task have many subsections and each individual core will work on each sub-sections. This means processor with many cores can work many of those subsections and once using parallel processing, allowing the task to be complete faster compared to sequential processing.

(5p) Identify the basic form of data and task parallelism in computational problems.

The basic form of data parallelism in computational problems is when same computation is applied to multiple data items. When you have a lot of data that you want to process, you divide it among multiple processors. Task parallelism is organized around the functions to be performed rather than the data. There are multiple tasks that need to be done. Different processor can be assigned for each task. So, the different processors will each look at the same data set.

(6p) Explain the differences between processes and threads.

A process is when a program runs. When a process starts, it is assigned memory and resources. A thread is a unit of execution within a process. It breaks the process down into smaller independent pars. Each thread in the process shares the process's memory and resources. Each process has separate memory address space.

(3p) What is OpenMP and what is OpenMP pragmas?

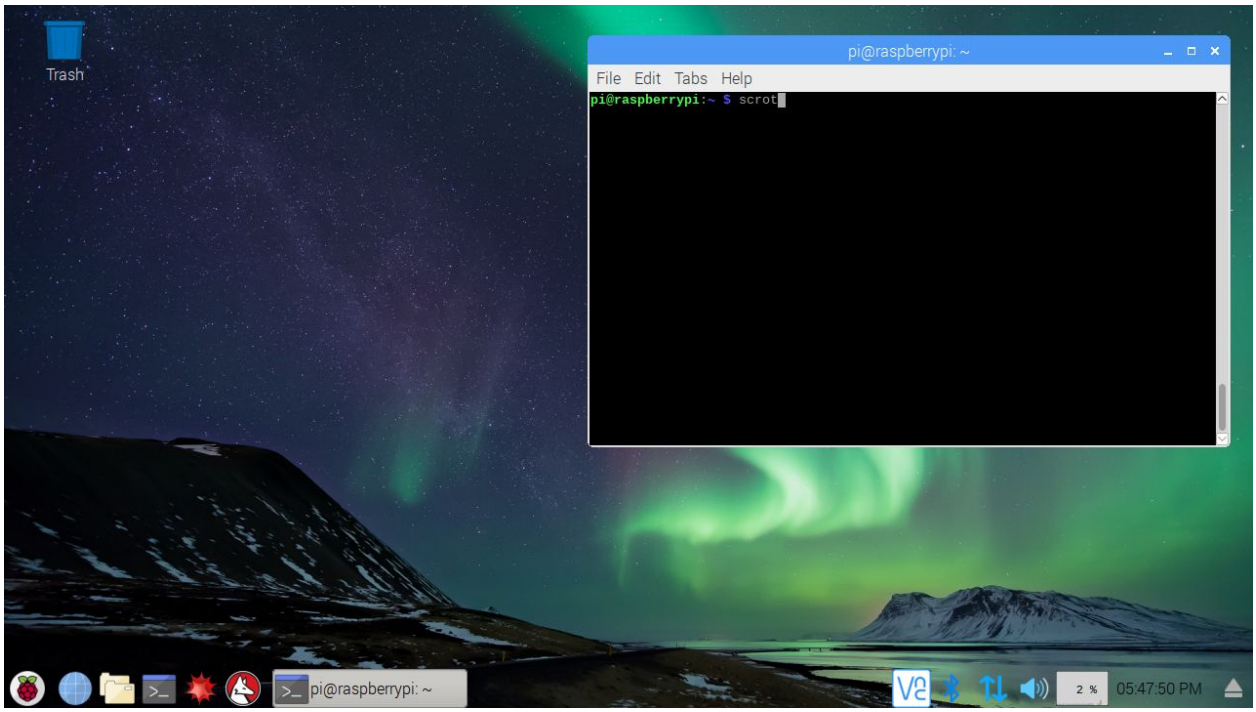
OpenMP or Open Multi-Processing is an API for C, C++, and Fortran that deals with multi-core process programming. OpenMP has a simple and flexible interface. Open MP pragmas are compiler directives. Compiler directives specify how a compiler should process its input. Pragmas are an alternative to pthreads. Pthreads explicit multithreading model requires the programmer to explicitly create and manage threads. In OpenMP pragmas implicit multithreading model, the library handles thread creation and management.

(4p) What applications benefit from multicore (list four)?

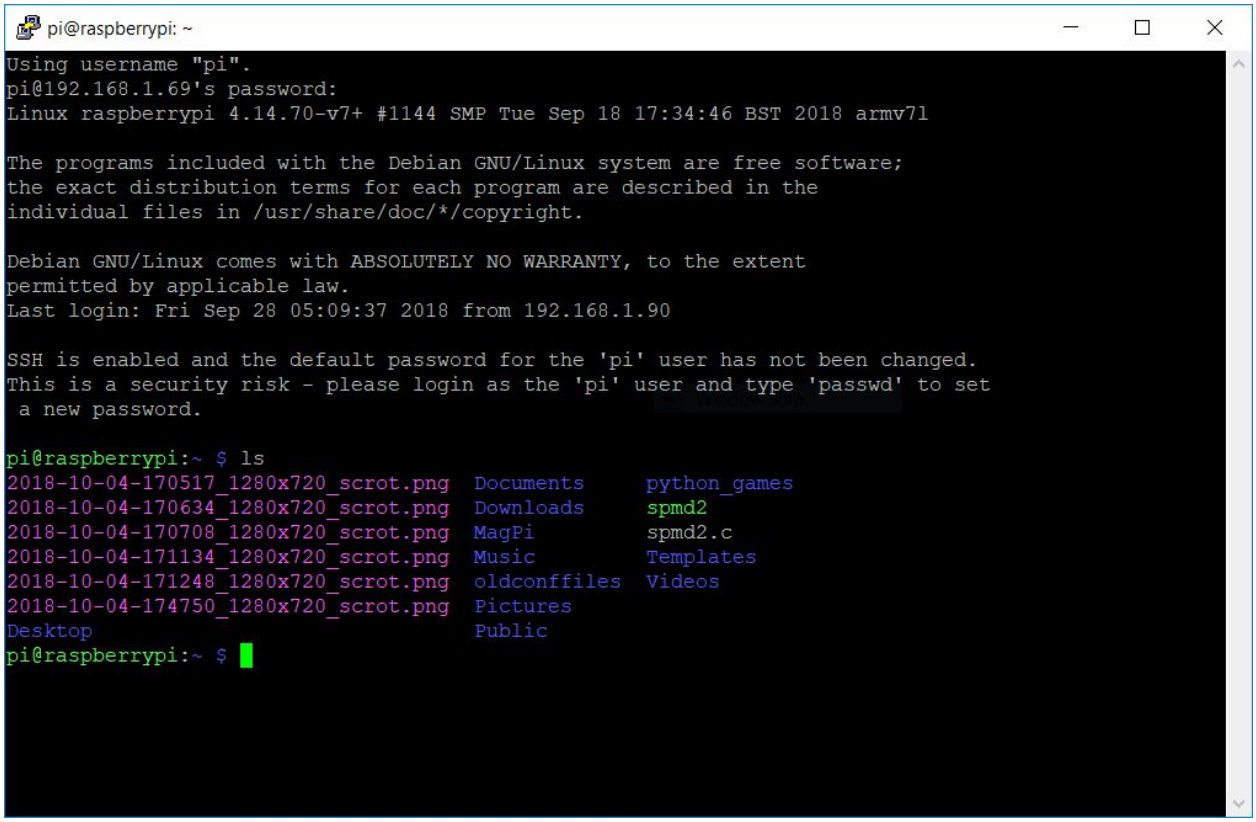
1) Photo and video editing software such as Adobe Photoshop and Adobe Premiere
2) Graphics-intensive games
3) 3-D Modeling
4) Server/Virtual Machine

(4p) Why Multicore? (why not single core, list four)

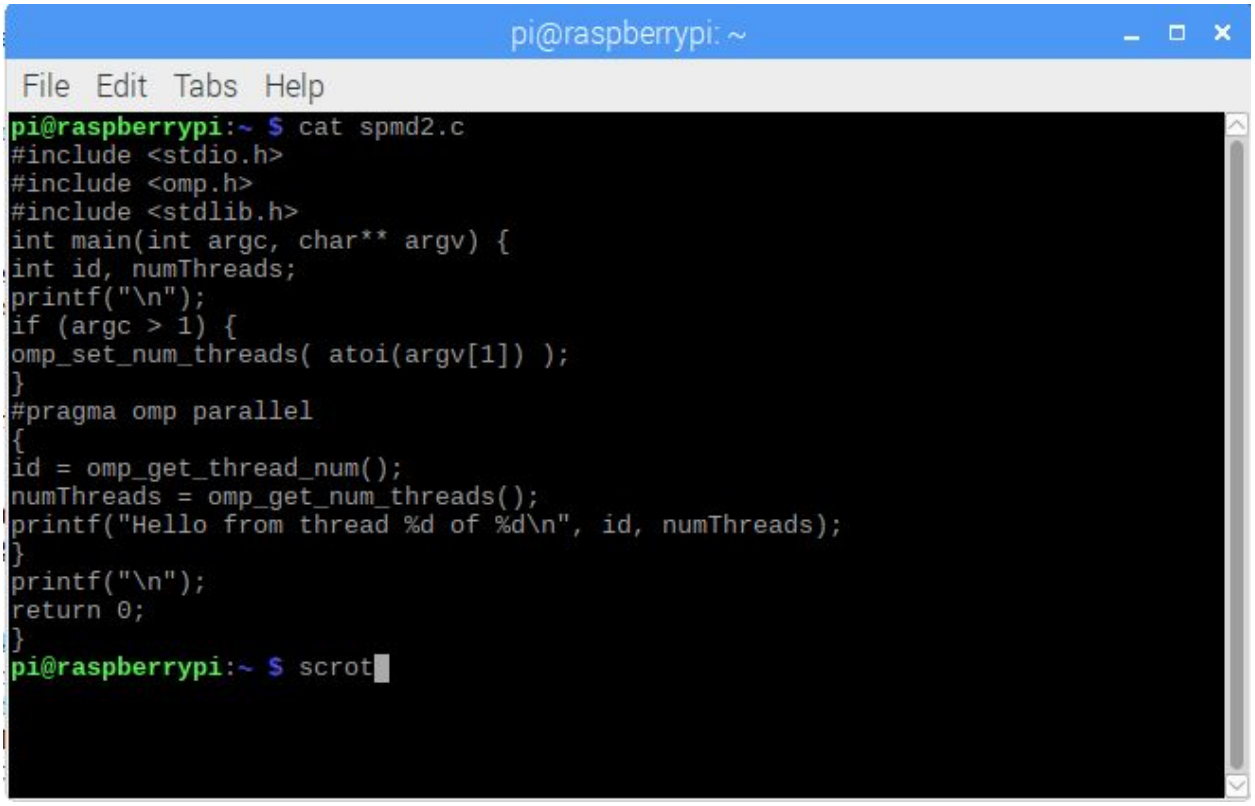
- Virtualization: Multicore allows you to dedicate individual cores to each virtual machine. The additional memory channels with more cores can handle demanding application in VM.
- Databases need many tasks to run simultaneously. Multicore allows multiple tasks to be run at once because there is increased memory.
- Multicore is better for high-performance computing. Since high performance computing is the practice of taking solving complex computations, each piece can be solved by different cores.
- Multiple cores improve overall processing performance. Since multiple instructions can be run on separate cores at the same time, overall speed increases.



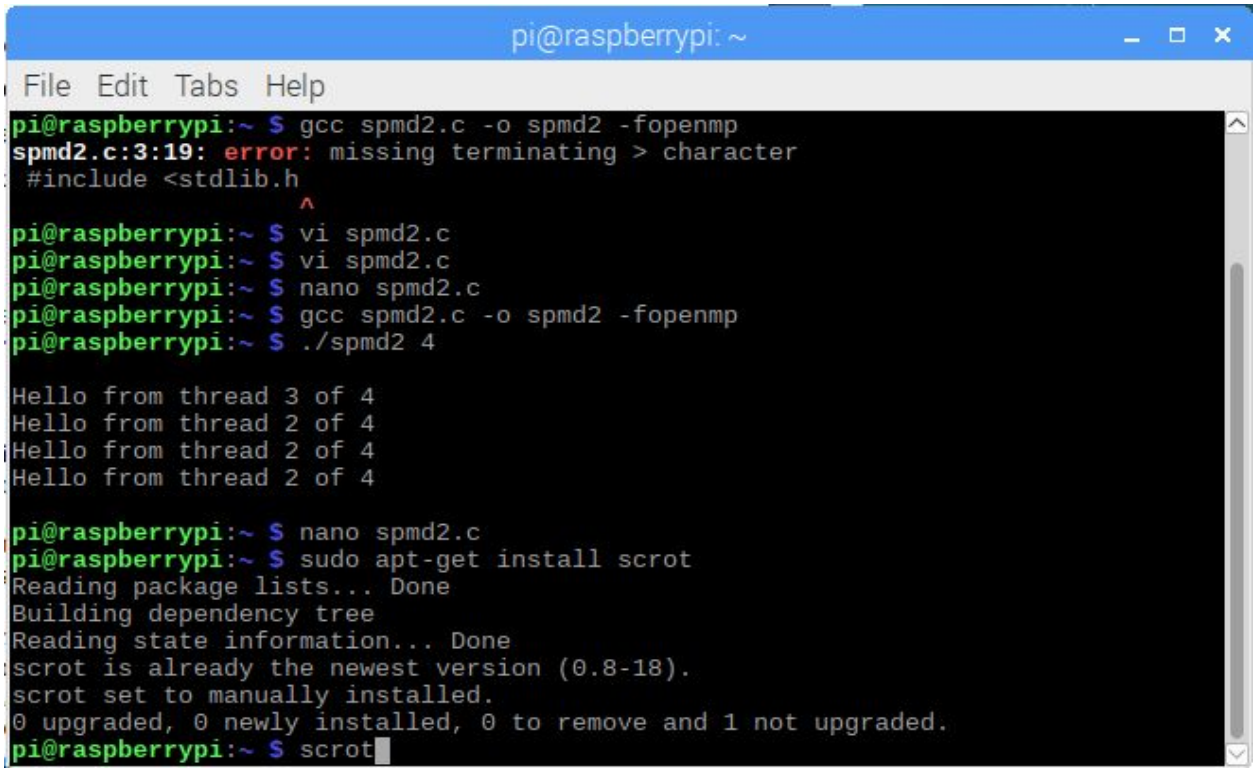
Setting up the Raspberry Pi with monitor, keyboard, and mouse was the easiest part. After installing the OS, all need to be do to complete the set up was to plug in the Pi into monitor with HDMI cable and follow the step until landing onto desktop screen.

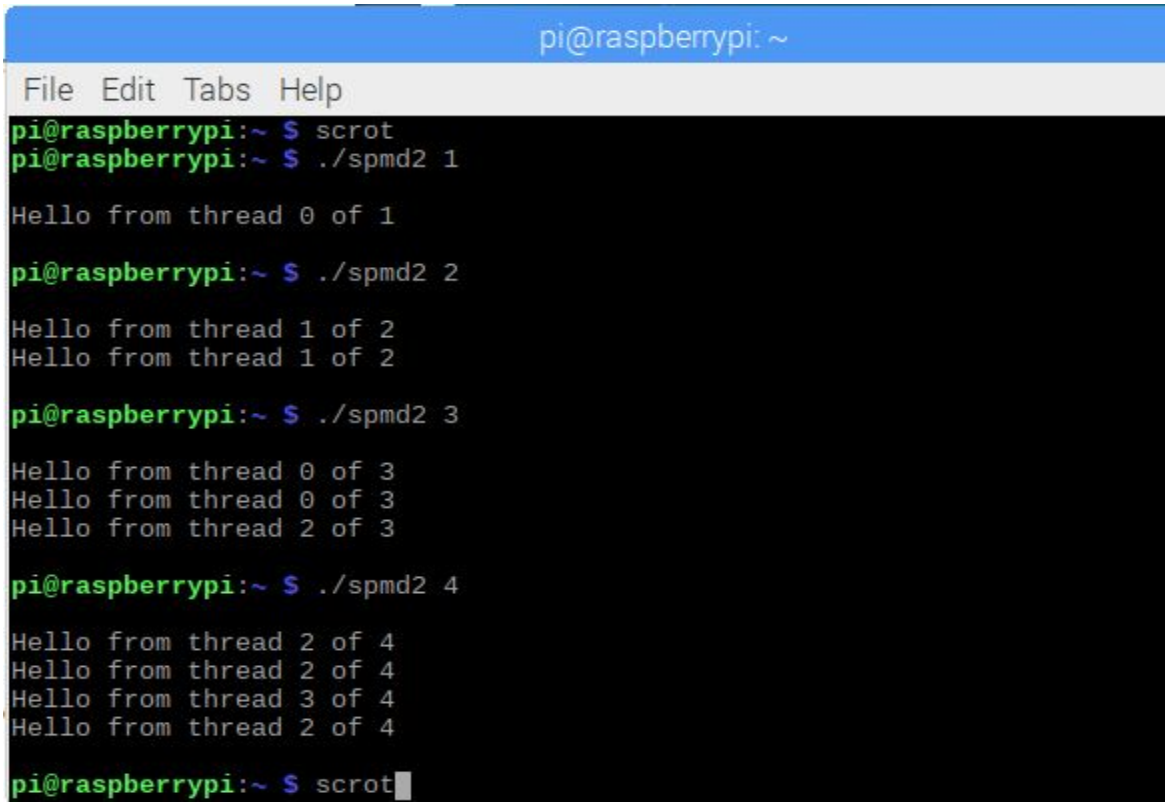


However, connection Pi with laptop was hardest part as the instruction provided on icollege was either little bit off or lack in details. Nonetheless, we managed to connect it with a laptop. After going through some of tutorial and resources online, it was rather simple as enabling SSH connection from Raspberry Pi's preference page and then connecting with right IP address.

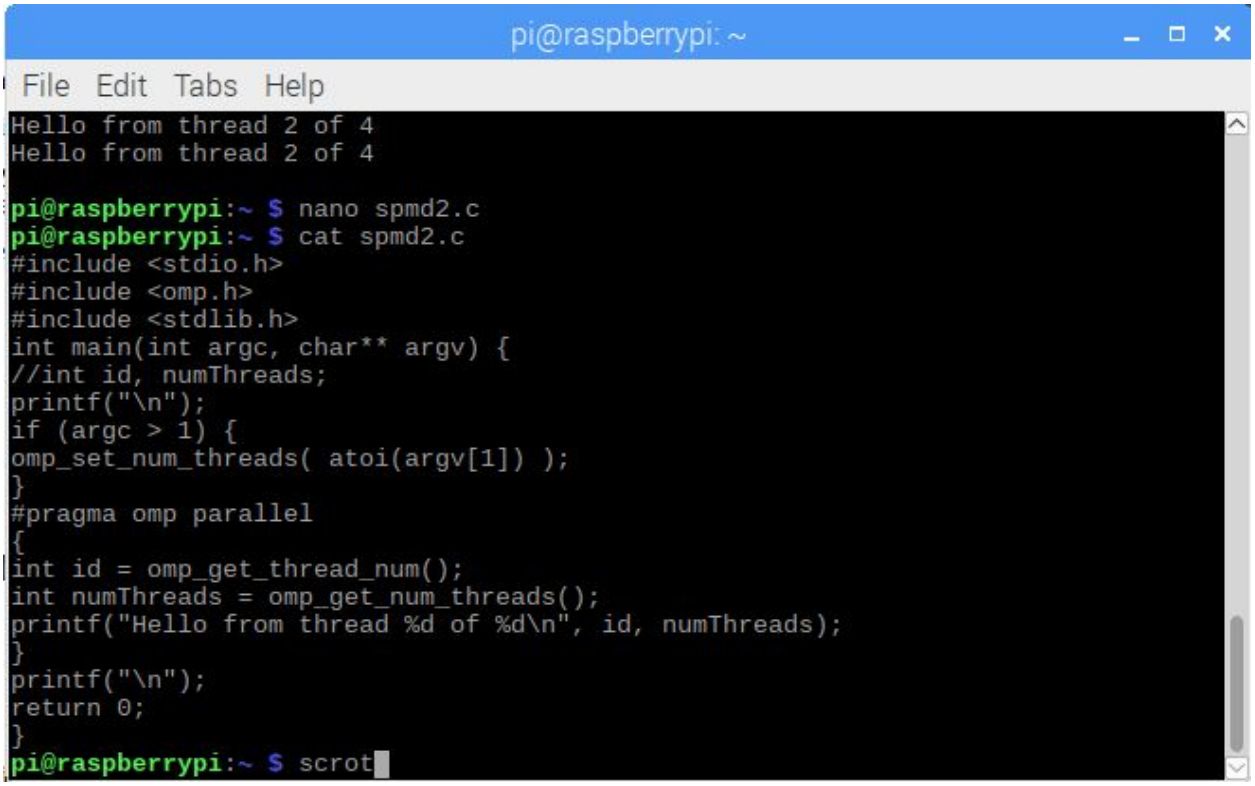


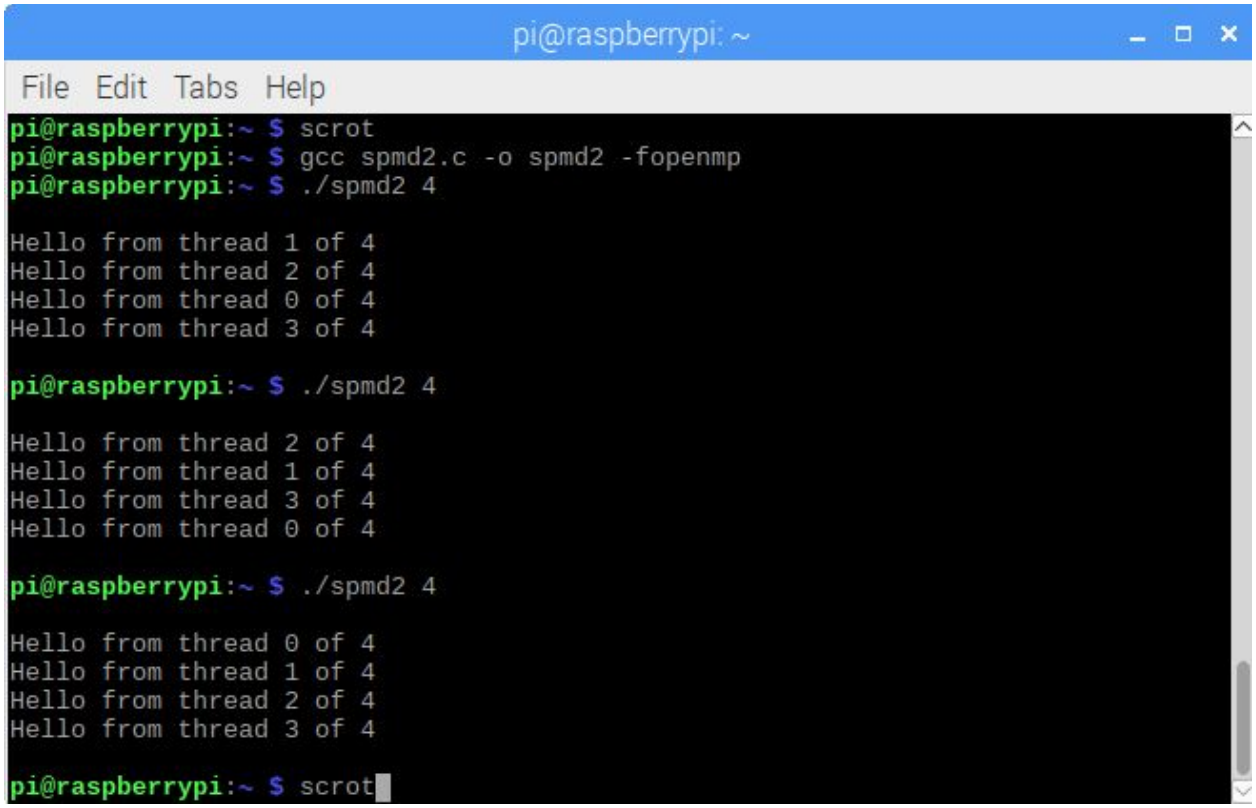
The first code above seemed both familiar and foreign to all of us due to our lack of knowledge in C. Our group members were familiar with the syntax since Java and C share many syntax similarity, but otherwise it was very new to us.





After writing the C code for the first time with Pi, we try to compile the program, but we ran across a mirror error. It was syntax error where we forgot to put one of “>”. After fixing the syntax error, the code compiled and ran. However, we knew something was wrong. This code is supposed to be a parallel programming meaning all core need to be utilized, but the result showed only 2 cores, at max, were doing the work. After running the code few more time, we were definite on the issue. Something was wrong, but didn’t know what.





As we follow the instruction, the second code showed us to undo the global initialization of variable ‘id’ and ‘numThread’ and initialize them individually. At first, we thought this would not make any difference. However, after compiling and running the code, it made the code to

work as intended. After running the code couple of times we were sure all the cores were doing their portion of the work, each core processing the string once.

Appendix:

Slack group link: https://madscientistsgsu.slack.com
YouTube channel link: https://www.youtube.com/channel/UCwX3csMyKmIZLRPmLBuAk5Q
GitHub link: https://github.com/MadScientists3210/CSC3210-MadScientists-