Project 1 - Add syscall to linux kernel on raspberry pi
By: Julio Anacleto && Abdullah Almaroof

Hardships / Challenges faced
- Large amount of time spent trying to cross compile kernel across home server network, ultimately built the kernel natively and found the process much easier.
- Confusion around the implementation of two kernel trees being used at the same time. Realized that our configurable kernel will be the main kernel we will boot into and the original will not be built and instead used to create patch files using the diff command.
- Struggled to find a comprehendible set of instructions for adding a syscall to our kernel and proceeding to call said syscall with a userspace application.
- The tutorial we found was intended for a linux kernel v4.13.0 while our current linux kernel was v6.12.51. This lead to difference between instructions and what we ultimately did as we gained assisstance from other class mates that were also working to configure the set of instructions to our current kernel.
- Found that the instructions had listed a syscall in the userspace application that did not exist in our version of the linux kernel which resulted in our userspace application returning a -1 when ran. We went through the system map trying to diagnos what our issue was and ultimately found that we were attempting to call a syscall that did not exist in our kernel.
- Encountered a set of warning that were generated from not having a line of space between our syscall in the syscall.h header file. This proceeded to generate a variety of warnings that might have caused issues and so we attempted to resolve this before finishing the build process.
- Attempted to use github through the rapsberry pi push our version of the linux kernel onto a github reposittory so that we could easily access files that we would use to upload to canvas for a final submission. Spent two hours attempting this procedure but continued to run into a road block with authentication for access to the github repository.
- Continued to make mistakes in our completed versions of the linux kernel and ultimately had to rebuild a new kernel repeatedly.
- Tried to cross compile (Save yourselves)

Contributions
Abdullah:
- Found tutorial used to create system call
- Corrected mistakes in altered files
- Researched alternative methods

Julio:
- Created linux kernel that is being submitted
- Made sure team had two kernel trees
- Researched alternative methods

Dmesg results

```
julio@raspberrypi:~ $ dmesg | tail
[    3.937350] macb 1f00100000.ethernet eth0: configuring for phy/rgmii-id link mode
[    3.940251] macb 1f00100000.ethernet: gem-ptp-timer ptp clock registered.
[    3.968622] brcmfmac: brcmf_cfg80211_set_power_mgmt: power save enabled
[    4.608343] tun: Universal TUN/TAP device driver, 1.6
[    5.949745] Bluetooth: RFCOMM TTY layer initialized
[    5.949761] Bluetooth: RFCOMM socket layer initialized
[    5.949771] Bluetooth: RFCOMM ver 1.11
[  193.922535] warning: `ThreadPoolForeg' uses wireless extensions which will stop working for
Wi-Fi 7 hardware; use nl80211
[  384.963914] Julio Anacleto & Abdullah Almaroof
[  384.963922] Hello world
```

Alternative Method:
Using the command "echo -e "<6>Abdullah Almaroof& Juilo\n<6>Hello World" | sudo tee /dev/kmsg > /dev/null" will display the samething as using the syscall that we implmented, this method is of
course a lot easier to implement as it is only one line that can be pasted into the terminal and then when "dmesg | tail" is run it will show the following:

```
admin@Almaroof-Pi5:~ $ echo -e "<6>Abdullah Almaroof& Juilo\n<6>Hello World" | sudo tee /
dev/kmsg > /dev/null
admin@Almaroof-Pi5:~ $ dmesg | tail
[    4.390465] Bluetooth: RFCOMM TTY layer initialized
[    4.390480] Bluetooth: RFCOMM socket layer initialized
[    4.390488] Bluetooth: RFCOMM ver 1.11
[    5.394715] macb 1f00100000.ethernet eth0: PHY [1f00100000.ethernet-ffffffff:01] driver
[Broadcom BCM54213PE] (irq=POLL)
[    5.394723] macb 1f00100000.ethernet eth0: configuring for phy/rgmii-id link mode
[    5.397565] macb 1f00100000.ethernet: gem-ptp-timer ptp clock registered.
[    5.425401] brcmfmac: brcmf_cfg80211_set_power_mgmt: power save enabled
[    6.453666] tun: Universal TUN/TAP device driver, 1.6
[   31.540581] Abdullah Almaroof& Juilo
               <6>Hello World
admin@Almaroof-Pi5:~ $
```

The echo command in a terminal is used to display lines of text or strings as output and when used with the <6>, it is able to output this string in the kernel info level by telling the kernel that this an informational log entry. The -e allows us to use the '\n' so we are able to output on a new line, the | (pipe) takes the echo command and writes this output to the kernel message log at
/dev/kmesg and normally it will also output this in the terminal but because we used the > /dev/null command it outputs the echo statement into a file that is discarded.