/proc/locks Covert Channel Guide
Written by: Benjamin Steenkamer, December 2017

Compiling:
In the directory `/proc-locks-no-ack`, type: `make`
Two binaries should be created: `sendWithLocks` and `receiveWithLocks`

Running:
`receiveWithLocks` should be placed in the container that will receive the data.
`sendWithLocks` should be placed in the container that will send the data.
Both containers must be running on the *same physical machine*.

Using some method, start both of these programs on their respective containers at the same time. If they are not started simultaneously, the sender might not find the receiving container and exit. In this case, the sender will say: `Receiver didn't send ACK. No data will be transferred.`
The receiver will also say:

```
Analysing lock list…
     X/9 valid locks were found.
Sender was not found.
```

Other issues can occur if the two programs are out of sync. Try to start both programs at the exact same time to avoid issues. This is difficult to do by hand, so I suggest writing a script to start both the programs at the same time for you. I use a bash script that forwards run commands to different `tmux` panels that display the two containers.

Configuration:
Currently, the sender is configured to only send 32 bytes. The number of bytes to be sent is defined in manageProcLocks.h. The defined constant is called `BYTES_TO_TRANSFER`. Change this value to the number of bytes you want to send. Then go into sendWithLocks.c at line 123 in the transfer_data function. There is an array that contains the bytes that will be sent. Increase the contents / length of this array to match the value of `BYTES_TO_TRANSFER`. This method is useful if you want to know, beforehand, what bytes you are sending.
OR ...
Instead of manually filling in the values into `dataArray`, you can comment out the first for-loop in the transfer_data function and uncomment the second for loop. This second for-loop will generate `BYTES_TO_TRANSFER` number of random bytes and send them. Right now this for-loop is the active one in the source code.
Regardless of the method used to generate the bytes, the current system uses only 8 data locks at one time (i.e. one byte is sent at a time). The number of data locks can be increased to allow more bits to be sent per "transmission," however the increased overhead from managing more than 8 locks can actually reduce the channel's bandwidth.