NOTE: THIS DOCUMENT IS A MESS. JUST ROUGH THOUGHTS

“Have our cake and eat it too”

TODO:
Run under strace (for functionality)
find.o for lkl and grepping through its nm output for the undefined symbols (in libc)

9/20/2019
Installing lkl. Cloned into lkl at https://github.com/lkl/linux.git. Tried to build lkl based tools for first time with:

```
$ sudo apt-get install libfuse-dev libarchive-dev xfsprogs

# Optional, if you would like to be able to run tests
$ sudo apt-get install btrfs-tools
$ pip install yamlish junit_xml

$ make -C tools/lkl

# To check that everything works:
$ cd tools/lkl
$ make run-tests
```

However, got an error:

```
HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
/bin/sh: 1: flex: not found
scripts/Makefile.lib:202: recipe for target 'scripts/kconfig/zconf.lex.c' failed
make[1]: *** [scripts/kconfig/zconf.lex.c] Error 127
Makefile:514: recipe for target 'oldconfig' failed
make: *** [oldconfig] Error 2
```

Fixed with:

```
apt-get install flex
```

During tests stuck at ‘setup_backend tap’ for a very long time. This could be due to my very old 32bit linux system.

Summary of tests: 18 suites run, 173 tests, 163 ok, 0 not ok, 10 skipped

Vde not supported.

Found lkl.o file as well as a vmlinux.o file. Do not really know what to do with it. Tried to do ./lkl.o got permission denied. Tried to do sudo ./lkl.o, got sudo: ./lkl.o command not found.

Started to look into makefile. Saw that its a recursive build. Read into the different types of build commands: i.e: cd /dir/to/output/files; make -f dir/to/kernel/source/makefile. We can also save files using make O=dir/to/output/file

Use ‘make c=1’ to enable checking of re-compiled files

Use ‘make c=1’ to enable checking of *all* source files

Important section:

Cross compiling and selecing different set of gcc/bin-utils

Note: make file super long a lot longer than composite around 2000 lines I wonder why.

Cleaning can be done on 3 levels:

Make clean Delete most generated files

Make mrproper delete the current configuration and all generated files

Make distclean Remove editor backup files, patch leftover files etc.

Use make -s for more info i.r image name etc

Thought: The test list looks like a cool list comprehensive list of things that lkl does. Could the list of tests be what lkl “does” if so we could see if composite needs it. Solves strace problem?

Question: Does gabe mean building lkl with all the commands (strace) or running lkl with commands (strace).

Question: how do I grep a lkl.o file see above problems

Reading: Started looking through lkl documentation. Looked at features… really easy to get lost. I should ask about what is critical to read in order to find what I want to find.

TODO: Read more documentation. Ask above questions. Look at other projects that have used lkl i.e Unikraft.

9/22/2019

Tried to see whats in lkl.o

Tried objdump lkl.o got: File format not recognized.

Need to build a bigger context.

What is the linux kernel?

Memory management: Keep track of how much memory is used to store what, and where

Process management: Determine which processes can use the CPU, when, and for how long

Device drivers: Act as mediator/interpreter between the hardware and processes

System calls and security: Receive requests for service from the processes

What is the linux kernel library?

Is the kernel the same as the OS?

The OS has a shell and a kernel.

9/23/2019

I've reached a roadblock on what to do with lkl.o.

In the meantime I have tried to read up more on lkl and what it can be used for. I need to address the question “How does lkl make calls out to interact with the broader system?” and more specifically how do I call a function in lkl. Obviously my main goal is to use it in composite. However, I am trying to brainstorm an intermediate idea…

I have been looking into Unikraft, which is a system for building specialized OSes and unikernels tailored to the needs of specific applications. Which implements lkl as an external library. I have been reading that lkl serves as an arch port. I am a little unsure about how to do something without just straight up building my own OS. Most of the hackable systems that I know are linux distros so it would be dumb to add lkl to something that already uses the linux kernel.

I also worry that there is going to be a point in my research where I can not dance around the core concepts covered in OS, and will have to spend majority of my time self teaching myself those concepts (just to relearn again). I am not saying thats a bad thing but it will be a major time drain.

9/26/2019

I have made a little bit of progress I was able to check the file type of lkl.o and saw that it was a data file type. Through that I was able to use hexdump and now i have a more readable file.

Notes from meeting with Gabe 9/26/2019:

- Lkl.o is kind of a mess and we should not mess with it
- We should look at hijack instead
-

9/27/2019

This week I am going to try to annotate more of the documentation. I have also joined the slack so I think I can build a lot of context from there.

One of my problems I was having in annotating the pdf is the idea of what a primitive system data type is

I have more context as to what primitive system data types are and how they interact with the system. Gabe put it this way:

The lkl system can be like a layered system:

-Application
-Libc
-lkl
-native operations (primitive system data types that lkl depends on)
-linux kernel

The native operations provide abstractions that linux usually relies on the hardware for. But instead of using the hardware directly, they try to use normal linux operations to accomplish the same.

Ex:

An important function that the kernel that the kernel relies on is the ability to “disable interrupts” (cli) and “reenable interrupts” (sti). (hardware instructions) but at user-level (in the application process), you can’t use those instructions as they are sensitive i.e only the kernel can you use them.

So you have to instead use pthread_locks which are abstractions that any applications can use.

So the native operation would be to provide the function of disabling “interrupts” via the pthread_ functions.

We will be replacing native operations with cos operations.

I have more context so I decided to switch back to actually trying to lkl to run. It does not turn out very well. I have tried both.

One thing that I have found that might be promising is that in the make setup on lkl github it talks about a json file and I was looking at the scripts in. linux/tools/lkl/tests/hijack-test.sh

I think I actually found something… I used the hijack bash script and it seems to be running actual tests and some seem to be succeeding. I dont really know what tests I am running but it seems to be a lot of network stuff. I think this is because we want the network device drivers. I think that means we might be going somewhere! There were a lot of errors however, like can not find a netserver execitable.

I have started searching for some of these native operations and I think that some of them are in system map ex: dummy_clock sched_clock_running

I think this might be just random linux things.

We want to find lkl host operations w/ lkl ops and that is of type lkl_host_operations. With grep -r lkl_ops . which led to grep -r lkl_host_operations .