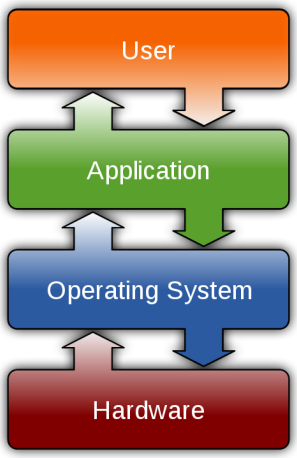
Let's Build a Kernel - Postmortem

Week 3

What *really* is an Operating System?

A collection of logic which support a computer's lowest level of functionality. The depth of functional responsibility of a given operating system is dependent on its intended purpose. In this way, an operating system can be tailor-made to run optimally for a given task, or made generic enough to support a wide variety of applications.



Why is the Kernel important?

At the heart of the operating system, you find the kernel. Compiled code with the purpose of controlling the flow of access to the computers resources (CPU, RAM, I/O, etc.).

Ubuntu is an operating system (distribution), which uses a Linux kernel

By optimizing our operating systems and their kernels, we can create optimal deployments which operate faster and at a much cheaper cost.

Huh?

Imagine you are deploying 100 instances of a server image to Amazon's AWS.

A stock built image, running at 100% utilization will cost you $1464.00 USD a month. If you could optimize the kernel to cut down the usage even slightly, say 80%, you just saved yourself $292.00 USD.

This is a small-scale example using the smallest tier of AWS' EC2 instances. The larger the deployment, the greater the savings.

The Process - What Went Wrong?

While we always hope that deployments go off without a hitch; there is a reason that it is frowned upon to push a deployment or build on a Friday afternoon.

1. Linux Refresher

A vital skillset when in a system administration role is the ability to effectively work in a Linux based environment. How effective were you in your ability to quickly recall and use your previous knowledge?

Useful Commands

cd	Change directory
mkdir	Make directory
mv	Move file / directory
rm	Remove file / directory
touch	Create empty file
cat	Output contents of file
df	Display free disk space and show mount points

Useful Files

/etc/fstab	Auto mounting devices and their mount points

When you are not using something frequently, you will forget it. There are numerous resources at your fingertips (Google) to help get you back up to speed.

2. Git

A commonly used version control system used extensively throughout numerous industries were versioning of code is a priority. Git is a mature, and actively maintained open source project developed by Linus Torvalds (the creator of the Linux kernel). Git in the past has had limitations when it came to handling binary files, however recent developments have resolved this through the Git Large File storage system.

There are many different flavors of versioning systems, with their own benefits and downsides. Git became popular through its extensive use by Open Source and numerous free repository hosting options (GitHub).

Useful Options

clone	Clone a repository into a new directory
init	Create an empty Git repository
fetch	Get branches/tags from repository
pull	Get latest state of repository for current/target branch

Handy Guide: https://wiki.ubuntu.com/KernelTeam/GitKernelBuild

3. Device Management

Unintentionally the disk volume did not have enough space to compile the current settings for the kernel. While there were numerous possible solutions to the problem, each with their own merits, here are a few possible solutions to a problem that occurs a lot more then you think.

When you provision a server, you are often asked to select the disk space to allocate. How do you handle when it outgrows its current storage over the course of its lifetime?

Expand The Current Volume – The Hard Way

This would typically be the first approach to resolve this sort of issue. It is the most commonly done operation to resolve size issues, assuming that the drive isn't the root volume and the virtual machine can be shutdown. *In this case of the lab, it was inadvertently left as a root volume. Whoops? Making this approach less desirable because of the*

Mount Another Device – The Easy Way

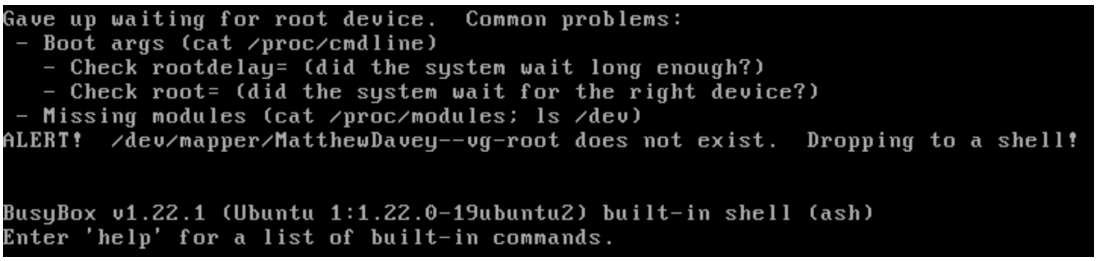
The simplest of solutions to the problem would be to add another hard drive into the mix, be it provisioned from a block storage provider (s3) or adding a physical drive should that be possible.

In this case of the lab, mounting the drive to the **/home/comp237/lab1** *folder would have provided a new unlimited space for the compilation to occur. The previous files would need to be copied over, assuming that the problem was realized after the download/compile had occurred*.

Strip The Kernel – The Masochist Way

When you compile the kernel, much like any other *make* system, an incredible amount of temporary files are created during the build process. The less amount of options and modules that are built into the kernel, the smaller the amount of space required is to build it. By removing unnecessary components, it would have been possible to build the kernel on the image without modifications to the images allocated space.

While we'll later delve into kernel stripping/optimization one of the inherit issues with removing parts of the kernel is creating a kernel that won't support the target hardware.



4. The Make System

The software compilation process can be a complicated mess, thankfully there are numerous build systems which simplify the process of identifying what files need to be included and establish the configurations necessary.

Copying Over The Previous Configuration

Unless you are extremely well versed in configuring a kernel, one important step that shouldn't be overlooked when building a kernel from scratch is copying over the existing builds configuration. This ensures that the build is done with at least a working baseline.

```
# cp /boot/config-`uname -r` .config
```

The **uname** command outputs the operating system name, with the added **-r** flag outputting the release information. In the above command, we use the outputted string to identify the appropriately named configuration file to copy.

5. Configuring The Kernel

Modifying the kernels configuration is a delicate process which overtime developers' have simplified and created tools to aid in the process.

Updating The Config

While not always necessary, updating the configuration copied from the current build may be necessary. Thankfully, there is an automated process available through an argument.

```
# make oldconfig
```

Kernel Signature

Configuring the kernel signature can be accomplished by editing the configuration manually as well as command line options. However, there is a much more visual way to configure the kernel which provides real-time feedback on the configurations and their options.

```
# make menuconfig
```

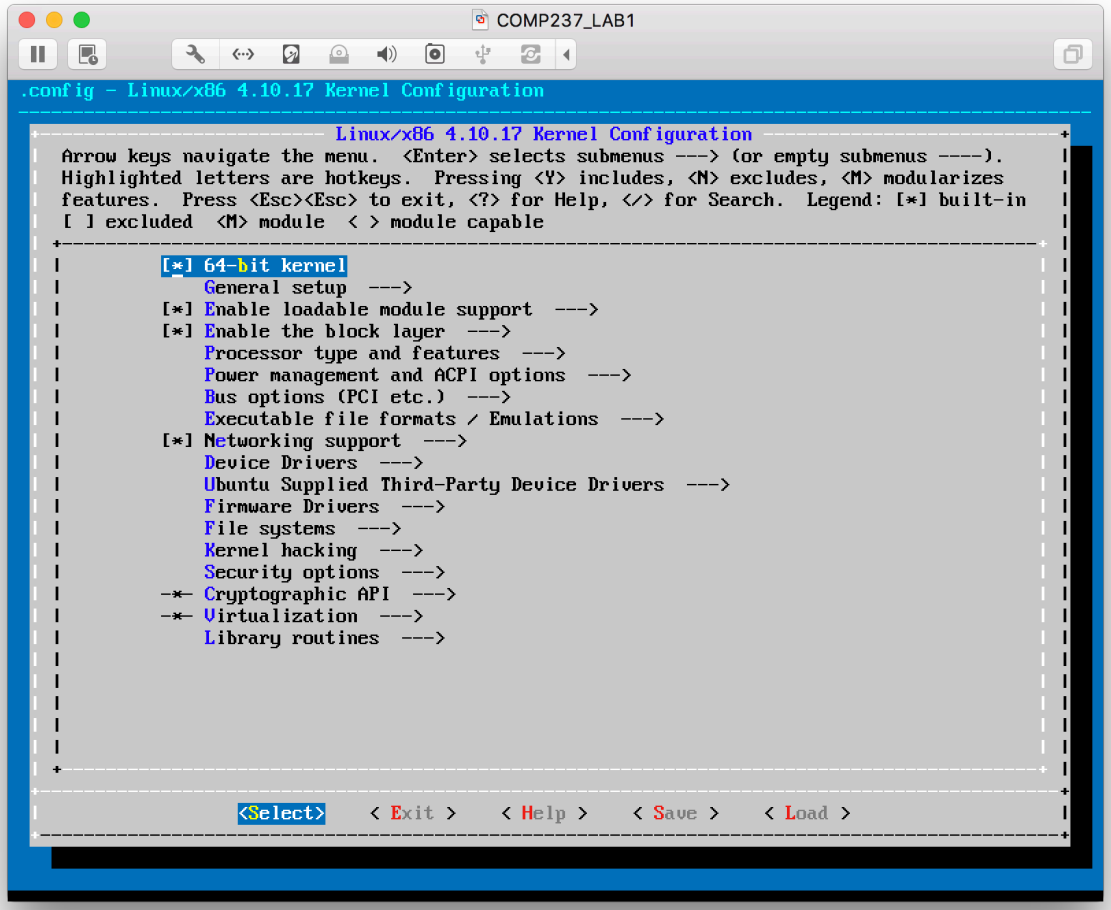


Figure 1 - The Kernel Signature can be found under General Setup

6. Building the Kernel

After having got this far, there is two remaining challenges in place. Building and installing the kernel. As mentioned prior this culmination of work could highlight prior problems and leave very little options other than rebuilding the image. Consider what would happen if you lost a machine remotely deployed because of a bad kernel.

Cleanup

As the build process makes all sorts of files associated with the build, it is often a good practice to clean to workspace between builds.

```
# make clean
```

Get It Done

The one command that will make or break the entire process. No pressure.

```
# make -j `getconf _NPROCESSORS_ONLN` deb-pkg LOCALVERSION=-myname
```

This will start the build process and output Debian packages to the parent directory above, named with the appropriate versioning.

A parent directory is the directory in which the current directory resides, similarly a child directory is a directory when referred to from its parent.

The Wait

It is real, go get a coffee, or start reading up on other topics.

7. Installing the Kernel

There is nothing better than seeing the successful completion of a long running compile. Having built the kernel, the final step is to install and restart to verify. A small hang-up often occurs not realizing that the output from the compilation is in the parent directory.

```
# dpkg –i *.deb
```

This will install the kernel into the appropriate places. One thing to be cautious of is that the command uses a wildcard filter and could potentially pick up other compiled packages.

Next Steps

If you were unable to finish the lab, use the above process to complete a kernel build.