Activity Two

Kernel Modules with eSDKs
Marco Cavallini
Kernel modules with eSDKs – Overview

• The Extensible SDK (eSDK) is a portable and standalone development environment, basically an SDK with an added bitbake executive via devtool.

• The “devtool” is a collection of tools to help development, in particular user space development.

• We can use devtool to manage a new kernel module:
  • Like normal applications is possible to import and create a wrapper recipe to manage the kernel module with eSDKs.
Kernel modules with eSDKs – Compiling a kernel module

• **We have two choices**

• **Out of the kernel tree**
  • When the code is in a different directory outside of the kernel source tree

• **Inside the kernel tree**
  • When the code is managed by a KConfig and a Makefile into a kernel directory
Kernel modules with eSDKs –
Pro and Cons of a module outside the kernel tree

- When the code is outside of the kernel source tree in a different directory

- **Advantages**
  - Might be easier to handle modifications than modify it into the kernel itself

- **Drawbacks**
  - Not integrated to the kernel configuration/compilation process
  - Needs to be built separately
  - The driver cannot be built statically
Kernel modules with eSDKs – Pro and Cons of a module inside the kernel tree

● When the code is inside the same directory tree of the kernel sources
● Advantages
  – Well integrated into the kernel configuration and compilation process
  – The driver can be built statically if needed
● Drawbacks
  – Bigger kernel size
  – Slower boot time
Kernel modules with eSDKs – The source code

```c
#include <linux/module.h>
#include <linux/kernel.h>

static int __init hello_init(void)
{
    printk("When half way through the journey of our life\n");
    return 0;
}

static void __exit hello_exit(void)
{
    printk("I found that I was in a gloomy wood\n");
}

guide_init(hello_init);
guide_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Greeting module from the Divine Comedy");
MODULE_AUTHOR("Dante Alighieri");
```
Kernel modules with eSDKs – The Makefile

obj-m += hellokernel.o

SRC := $(shell pwd)

all:
  $(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules

modules_install:
  $(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules_install

• **KERNEL_SRC** is the location of the kernel sources.
• **This variable is set to the value of the STAGING_KERNEL_DIR within the module class (module.bbclass)**
• **Sources available on https://github.com/koansoftware/simplest-kernel-module.git**
Kernel modules with eSDKs – Devtool setup

- **Start a new Shell!** Otherwise, the existing bitbake environment can cause unexpected results

- Here is how the eSDK was prepared for this class account:

  < DO NOT ENTER THE FOLLOWING COMMANDS : ALREADY EXECUTED >

  ```
  $ bitbake core-image-base -c populate_sdk_ext
  $ cd /scratch/working/build-qemuarm/tmp/deploy/sdk/
  $ ./poky-glibc-x86_64-core-image-base-armv5e-toolchain-ext-2.4.sh \
    -d /scratch/sdk/qemuarm -y
  $ cd /scratch/sdk/qemuarm
  $ . environment-setup-armv5e-poky-linux-gnueabi
  $ devtool modify virtual/kernel
  $$
  ```

- This installed the eSDK into:
  `/scratch/sdk/qemuarm`
Kernel modules with eSDKs – Overview

- Starting from now we are using the **eSDK** and not the project
- During this exercise we using two different machines
  - The **HOST** containing the eSDK (providing devtool)
  - The **TARGET** running the final qemuarm image
Kernel modules with eSDKs – Globalsetup

- Open two terminal windows and setup the eSDK environment in each one

```
$ cd /scratch/sdk/qemuarm
$ bash # safe shell
$ source environment-setup-armv5e-poky-linux-gnueabi
...
SDK environment now set up;
additionally you may now run devtool to perform development tasks.
Run devtool --help for further details.
```

Kernel modules with eSDKs – build the target image

• After you have setup the eSDK environment, build an image

$ devtool build-image

• This will create a new image into:

/scratch/sdk/qemuarm/tmp/deploy/images/qemuarm
Kernel modules with eSDKs – build the target image

- Run the image to check if everything is OK
- This will run the QEMU machine in the TARGET shell you were using
- Login using user: root (no password required)

$ runqemu qemuarm nographic
Kernel modules with eSDKs – Hooking a new module into the build

• Run the devtool to add a new recipe (on the HOST side)

```
$ devtool add --version 1.0 simplestmodule /
    /scratch/src/kmod/simplest-kernel-module/
```

• This generates a minimal recipe in the workspace layer
• This adds EXTERNALSRC in an workspace/appends/simplestmodule_git.bbappend file that points to the sources
• In other words, the source tree stays where it is, devtool just creates a wrapper recipe that points to it

• Note: this does not add your image to the original build engineer’s image, which requires changing the platform project’s conf/local.conf
After the add

Workspace layer layout

$ tree /scratch/sdk/qemuarm/workspace/

/scratch/sdk/qemuarm/workspace/
  ├── appends
  │   └── simplestmodule_git.bbappend
  ├── conf
  │   └── layer.conf
  └── README
      └── recipes
          └── simplestmodule
              └── simplestmodule_git.bb
Kernel modules with eSDKs – Build the Module

• **Build the new recipe** (on the HOST side)

```
$ devtool build simplestmodule
```

This will create the *simplestmodule.ko* kernel module

This downloads the kernel sources (already downloaded for you):
`linux-yocto-4.12.12+gitAUTOINC+eda4d18ce4_67b62d8d7b-r0 do_fetch`
Kernel modules with eSDKs – Deploy the Module

• Get the target’s IP address from the target serial console

  root@qemuarm:~# ifconfig

• In the eSDK (HOST) shell, deploy the output
  (the target’s ip address may change)

  $ devtool deploy-target -s simplestmodule root@192.168.7.2

• NOTE: the ‘-s’ option will note any ssh keygen issues, allowing you to
  (for example) remove/add this IP address to the known hosts table
**Kernel modules with eSDKs – Deploy Details**

- **In the target (qemuarm), observe the result of deployment**

<table>
<thead>
<tr>
<th>Command</th>
<th>Status</th>
<th>Size</th>
<th>Speed</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>devtool_deploy.list</td>
<td>100%</td>
<td>108</td>
<td>0.1KB/s</td>
<td>00:00</td>
</tr>
<tr>
<td>devtool_deploy.sh</td>
<td>100%</td>
<td>1017</td>
<td>1.0KB/s</td>
<td>00:00</td>
</tr>
<tr>
<td>./</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./lib/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./lib/modules/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./lib/modules/4.12.12-yocto-standard/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./lib/modules/4.12.12-yocto-standard/extra/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./lib/modules/4.12.12-yocto-standard/extra/hellokernel.ko</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./usr/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./usr/include/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./usr/include/simplestmodule/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./usr/include/simplestmodule/Module.symvers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./etc/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./etc/modprobe.d/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>./etc/modules-load.d/</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Successfully deployed

/scratch/sdk/qemuarm/tmp/work/qemuarm-poky-linux-gnueabi/simplestmodule/
Kernel modules with eSDKs – Load the Module

- **In the target** *(qemuarm)*, load the module and observe the results

```
root@qemuarm:~# depmod -a

root@qemuarm:~# modprobe hellokernel
[ 874.960165] When half way through the journey of our life

root@qemuarm:~# lsmod
Module    Size  Used by
hellokernel  929   0
nfsd      271348  11
```
Kernel modules with eSDKs – Unload the Module

- **In the target** (*qemuarm*), **unload the module**

```bash
root@qemuarm:~# modprobe -r hellokernel
[ 36.005902] I found that I was in a gloomy wood

root@qemuarm:~# lsmod
Module                      Size  Used by
nfsd                        271348  11
```
Kernel modules with eSDKs – automatic load of the module at boot

- In the target (qemuarm), edit the file below and add a new line containing the module name ‘hellokernelp’

```
root@qemuarm:~# vi /etc/modules-load.d/hello.conf

< insert the following line and save >

hellokernelp
```

- Then reboot the Qemu machine and verify

```
root@qemuarm:~# reboot
```
Questions