Working with SDKs

Sean Hudson

Embedded Linux Architect &
Member of Technical Staff

mentor embedded

mentor.com/embedded

Who am I?

- Embedded Linux Architect & MTS at Mentor Embedded, which is a division of Mentor Graphics
- Current member of the board for the OpenEmbedded Project
- Former representative to the Advisory Board for the Yocto Project

Outline

- What is a Yocto Project SDK?
- How to build
- How to use
- Final Thoughts

What is a Yocto Project SDK?

- A cross-compile toolchain
- A combination of **two** sysroots
  - One for the target
    - Contains headers and libraries for the target
    - NOTE: Consistent with the generated image from which it is derived
  - One for the host
    - Contains host specific tools
    - NOTE: These tools ensure things are consistent and work as expected while building against the target sysroot
- An environment script to setup the necessary variables to make these work together

So, what’s a Yocto Project SDK do?

- It allows a platform developer to provide a build environment to an application developer
- This environment is self contained with all the elements that an application developer needs to build an application on their host machine
- Enables the application developer to focus on developing their application
- Allows the platform developer to upgrade application developers entire build environment as desired

Where’s the code that handles this?

- There are a few classes that add the SDK function
  - populate_sdk_base.bbclass
  - populate_sdk.bbclass
  - populate_sdk_deb.bbclass
  - populate_sdk_ipk.bbclass
  - populate_sdk_rpm.bbclass

Variables that control the process

- IMAGE_PKGTYPE (PACKAGE_CLASSES)
- SDK_ARCH
- SDK_DEPLOY
- SDK_DIR
- SDK_NAME
- SDK_OUTPUT
- SDKIMAGE_FEATURES
- SDKMACHINE
- SDKPATH
- TOOLCHAIN_HOST_TASK
- TOOLCHAIN_TARGET_TASK

Note: These are well explained in the reference manual.

What about a Canadian Cross?

- Canadian Cross
  - Involves building a toolchain and sysroot on one host machine type for use on a different host machine type in order to compile for a target that is different from both hosts
  - Wikipedia link here

    http://en.wikipedia.org/wiki/Cross_compiler#Canadian_Cross

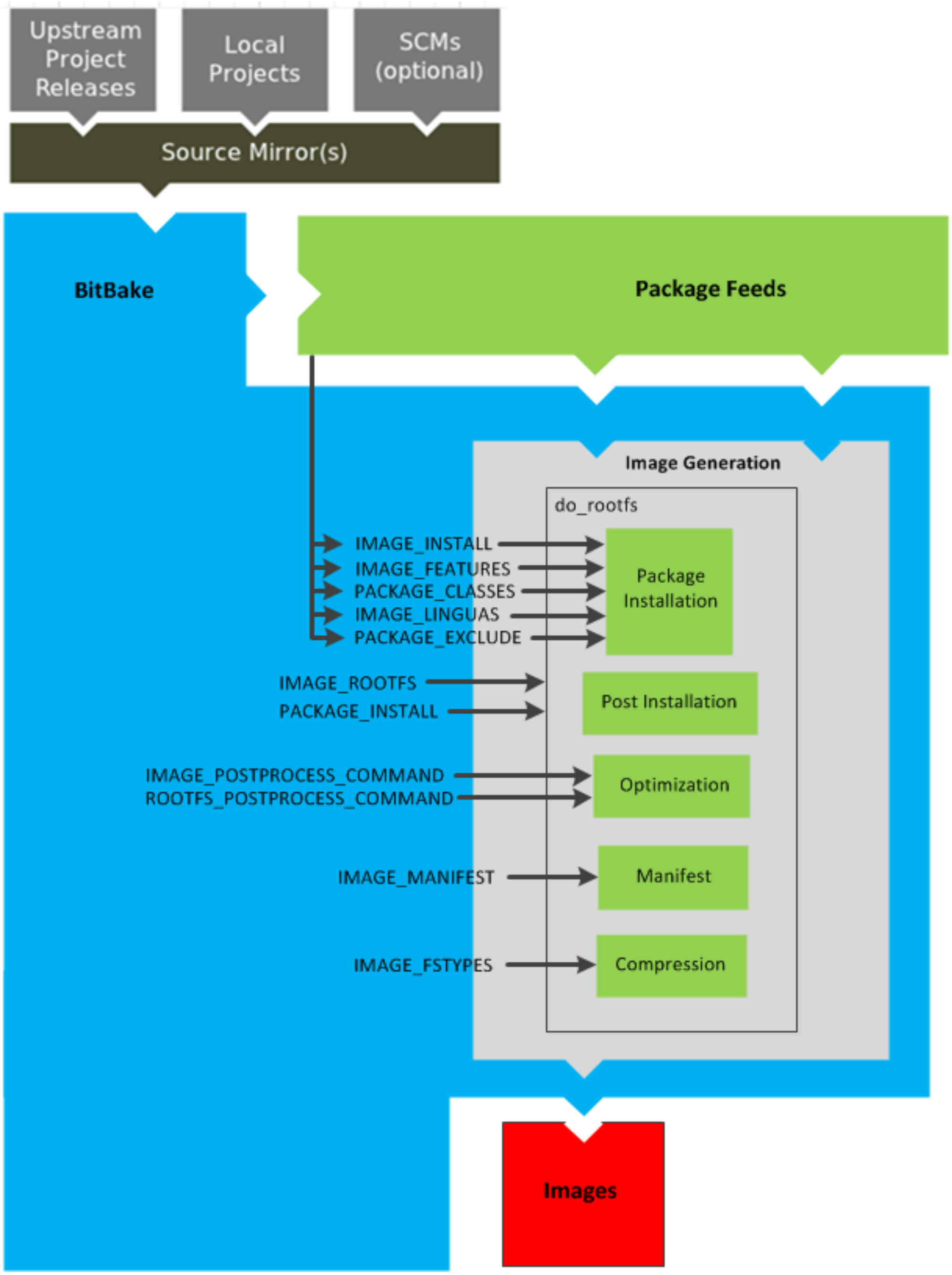
- SDKMACHINE variable controls the alternate host
  - Set this variable to the desired host
  - Works with x86 and x86-64 hosts

Building an SDK

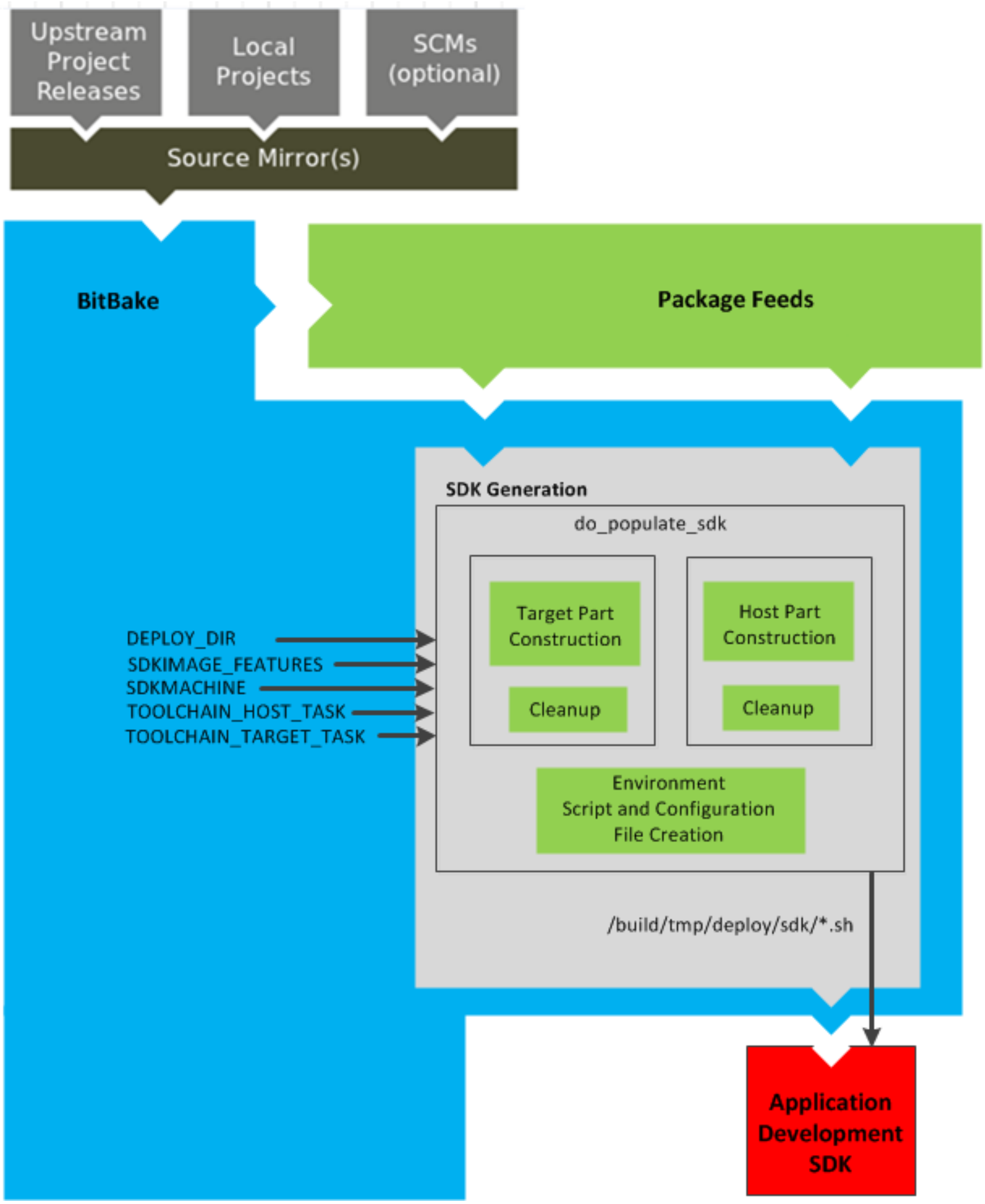
- Building an SDK is quite simple
- Just add “-c populate_sdk” to the bitbake command for an image
  - $ bitbake core-image-minimal –c populate-sdk

- Note: it is highly recommended that you build this in a clean tree

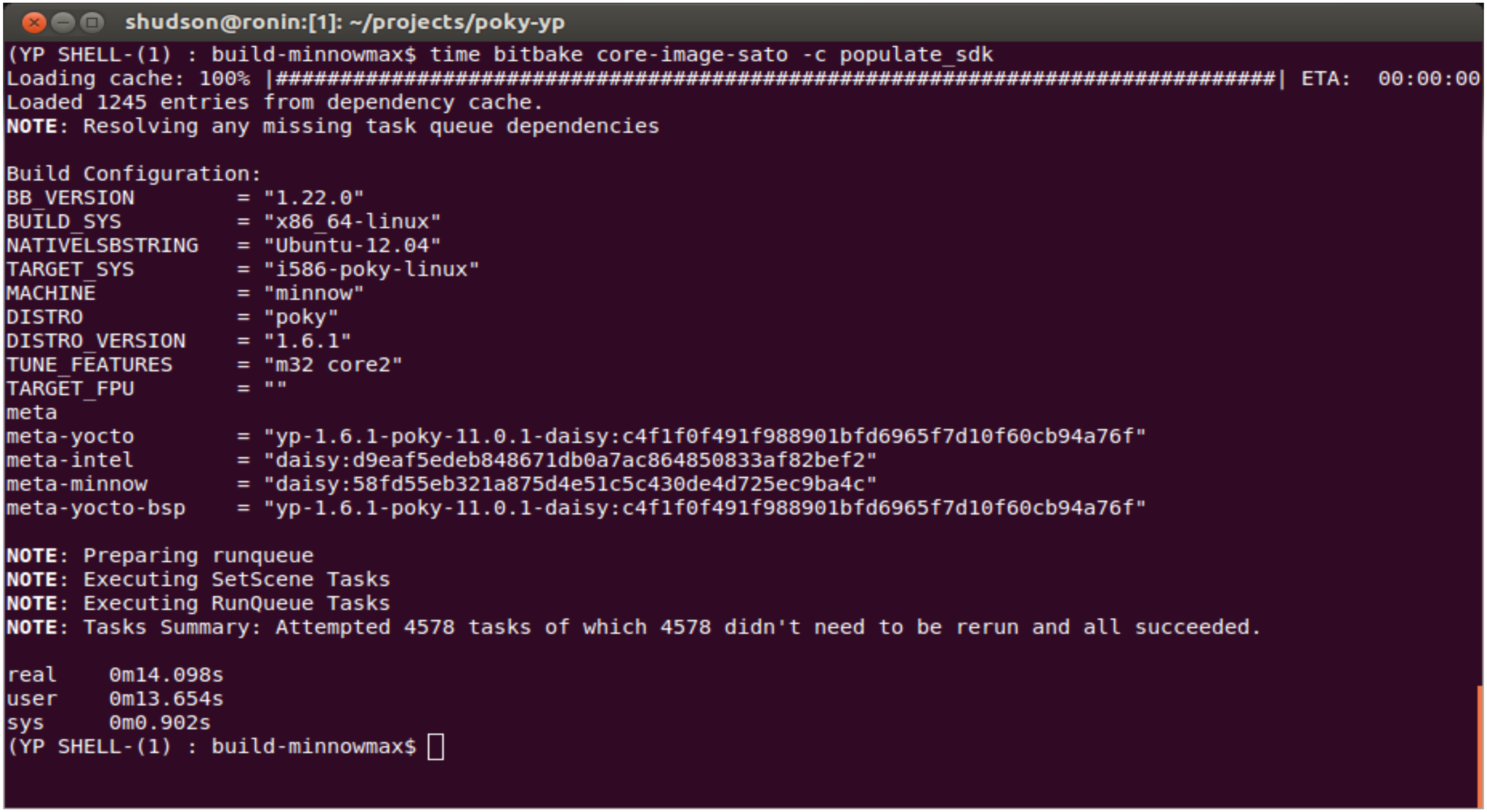
Building an Image



Building an SDK



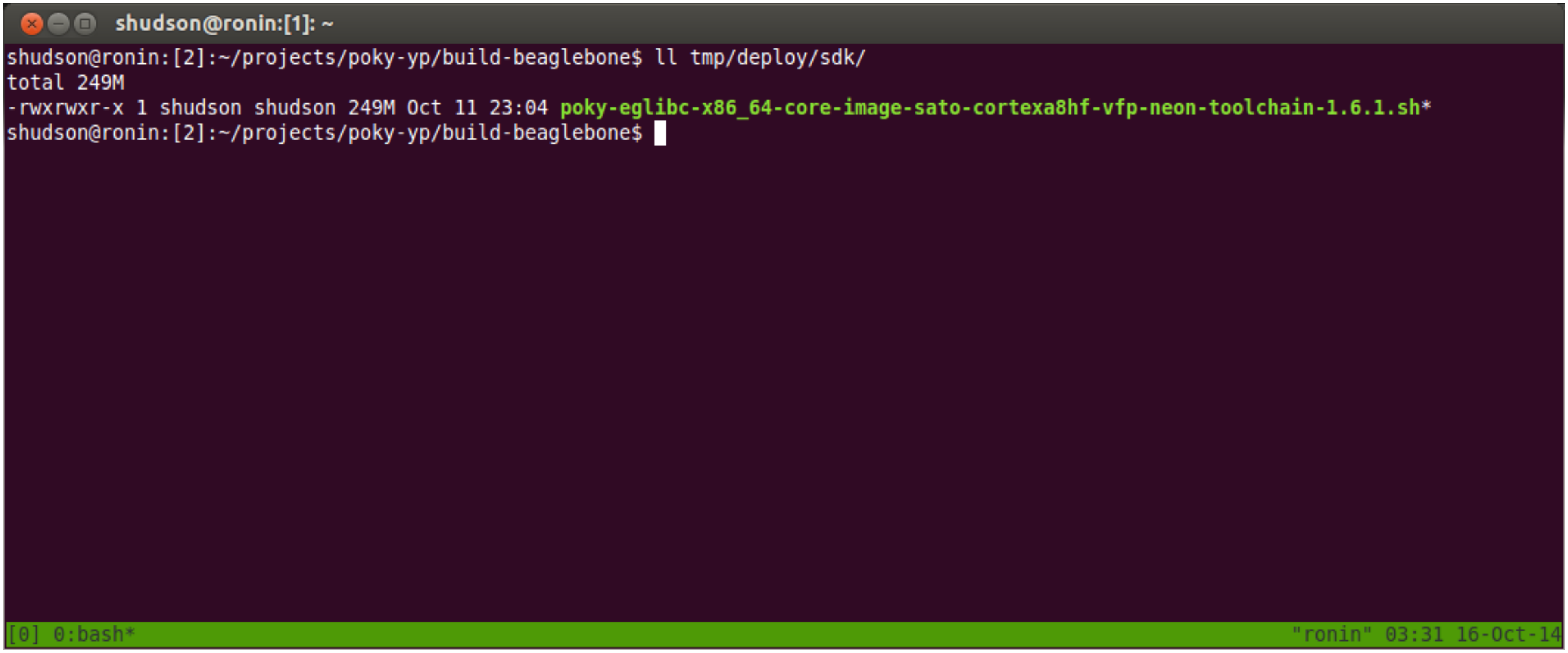
Build it!



```
(YP SHELL-(1) : build-minnowmax$ time bitbake core-image-sato -c populate_sdk
Loading cache: 100% |##############################################################################| ETA:  00:00:00
Loaded 1245 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION        = "1.22.0"
BUILD_SYS         = "x86_64-linux"
NATIVELSBSTRING   = "Ubuntu-12.04"
TARGET_SYS        = "i586-poky-linux"
MACHINE           = "minnow"
DISTRO            = "poky"
DISTRO_VERSION    = "1.6.1"
TUNE_FEATURES     = "m32 core2"
TARGET_FPU        = ""
meta
meta-yocto        = "yp-1.6.1-poky-11.0.1-daisy:c4f1f0f491f988901bfd6965f7d10f60cb94a76f"
meta-intel        = "daisy:d9eaf5edeb848671db0a7ac864850833af82bef2"
meta-minnow       = "daisy:58fd55eb321a875d4e51c5c430de4d725ec9ba4c"
meta-yocto-bsp    = "yp-1.6.1-poky-11.0.1-daisy:c4f1f0f491f988901bfd6965f7d10f60cb94a76f"

NOTE: Preparing runqueue
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
NOTE: Tasks Summary: Attempted 4578 tasks of which 4578 didn't need to be rerun and all succeeded.

real    0m14.098s
user    0m13.654s
sys     0m0.902s
(YP SHELL-(1) : build-minnowmax$ 
```

- Is that it?

Where did the SDK build output go?

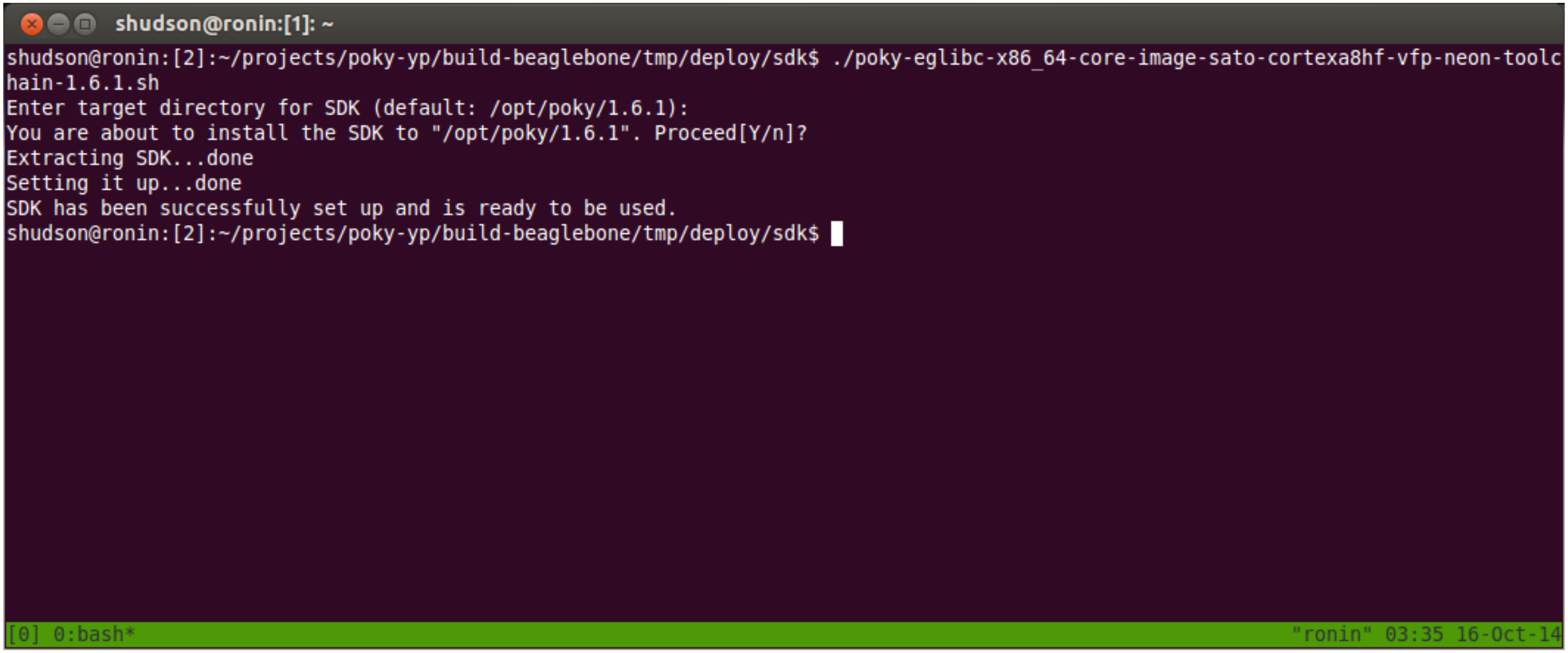


```
shudson@ronin:[2]:~/projects/poky-yp/build-beaglebone$ ll tmp/deploy/sdk/
total 249M
-rwxrwxr-x 1 shudson shudson 249M Oct 11 23:04 poky-eglibc-x86_64-core-image-sato-cortexa8hf-vfp-neon-toolchain-1.6.1.sh*
shudson@ronin:[2]:~/projects/poky-yp/build-beaglebone$ 
```

- One file?

Installing an SDK

- Execute the generated script



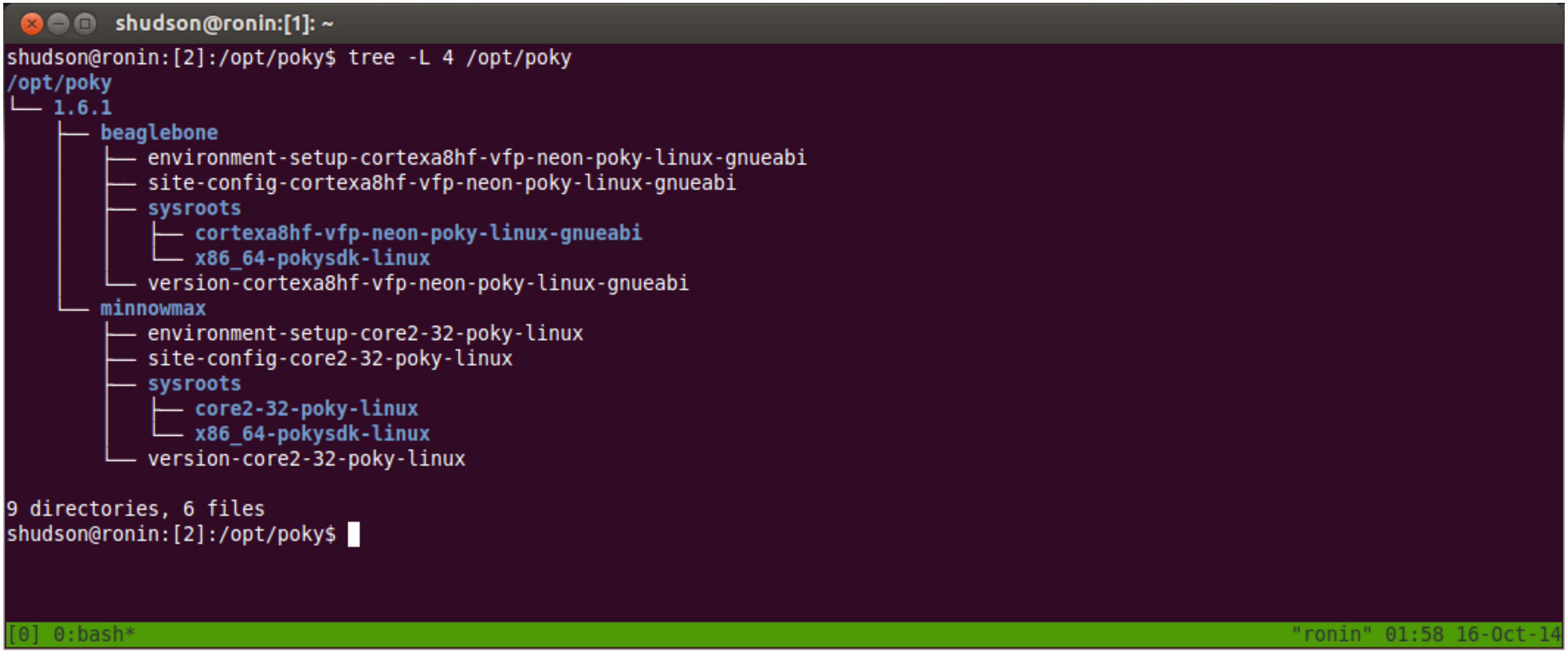
```
shudson@ronin:[2]:~/projects/poky-yp/build-beaglebone/tmp/deploy/sdk$ ./poky-eglibc-x86_64-core-image-sato-cortexa8hf-vfp-neon-toolc
hain-1.6.1.sh
Enter target directory for SDK (default: /opt/poky/1.6.1):
You are about to install the SDK to "/opt/poky/1.6.1". Proceed[Y/n]?
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
shudson@ronin:[2]:~/projects/poky-yp/build-beaglebone/tmp/deploy/sdk$ 
```

```
[0] 0:bash*                                                                                          "ronin" 03:35 16-Oct-14
```

- NOTE: this script automatically uses **sudo** to create the SDK install directory, if necessary

What just happened?

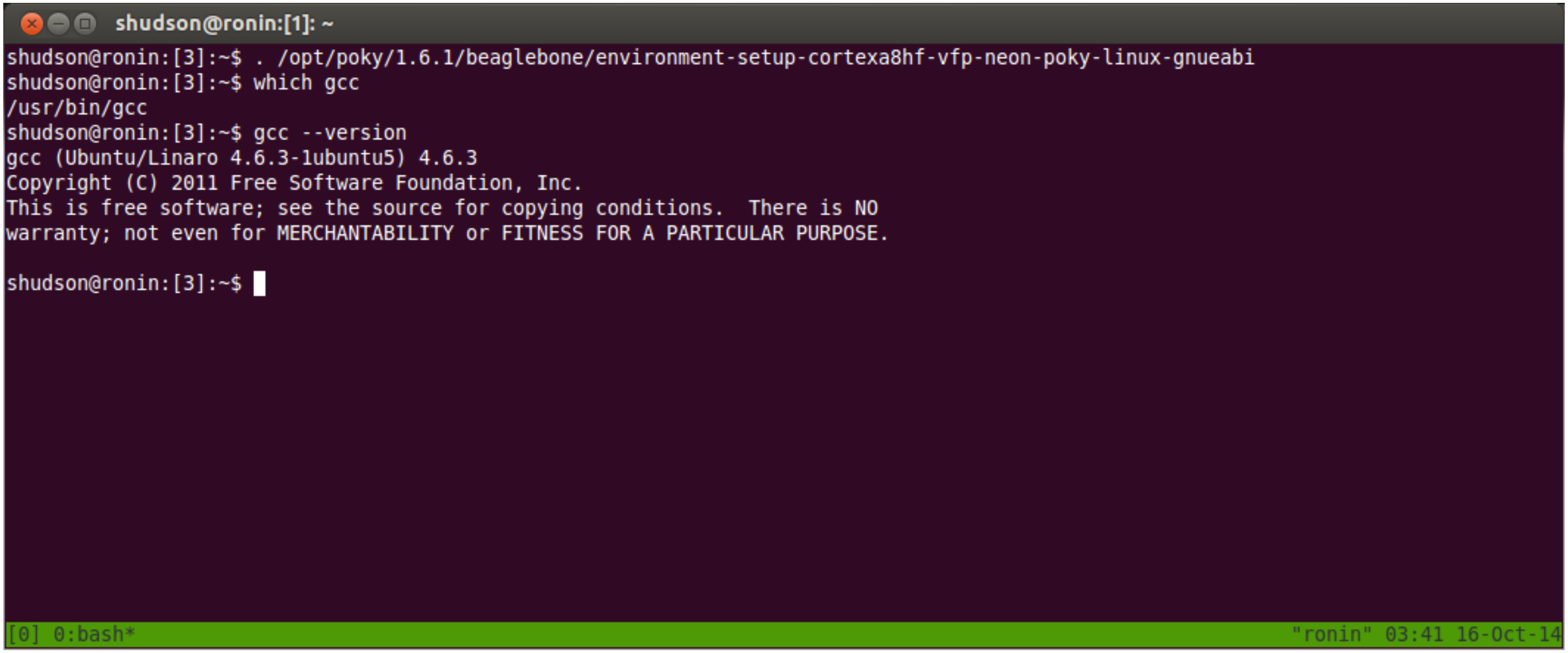
- Here's the top level of the installed SDK directories



```
shudson@ronin:[2]:/opt/poky$ tree -L 4 /opt/poky
/opt/poky
└── 1.6.1
    ├── beaglebone
    │   ├── environment-setup-cortexa8hf-vfp-neon-poky-linux-gnueabi
    │   ├── site-config-cortexa8hf-vfp-neon-poky-linux-gnueabi
    │   ├── sysroots
    │   │   ├── cortexa8hf-vfp-neon-poky-linux-gnueabi
    │   │   └── x86_64-pokysdk-linux
    │   └── version-cortexa8hf-vfp-neon-poky-linux-gnueabi
    └── minnowmax
        ├── environment-setup-core2-32-poky-linux
        ├── site-config-core2-32-poky-linux
        ├── sysroots
        │   ├── core2-32-poky-linux
        │   └── x86_64-pokysdk-linux
        └── version-core2-32-poky-linux

9 directories, 6 files
shudson@ronin:[2]:/opt/poky$ 
[0] 0:bash*                                                                            "ronin" 01:58 16-Oct-14
```

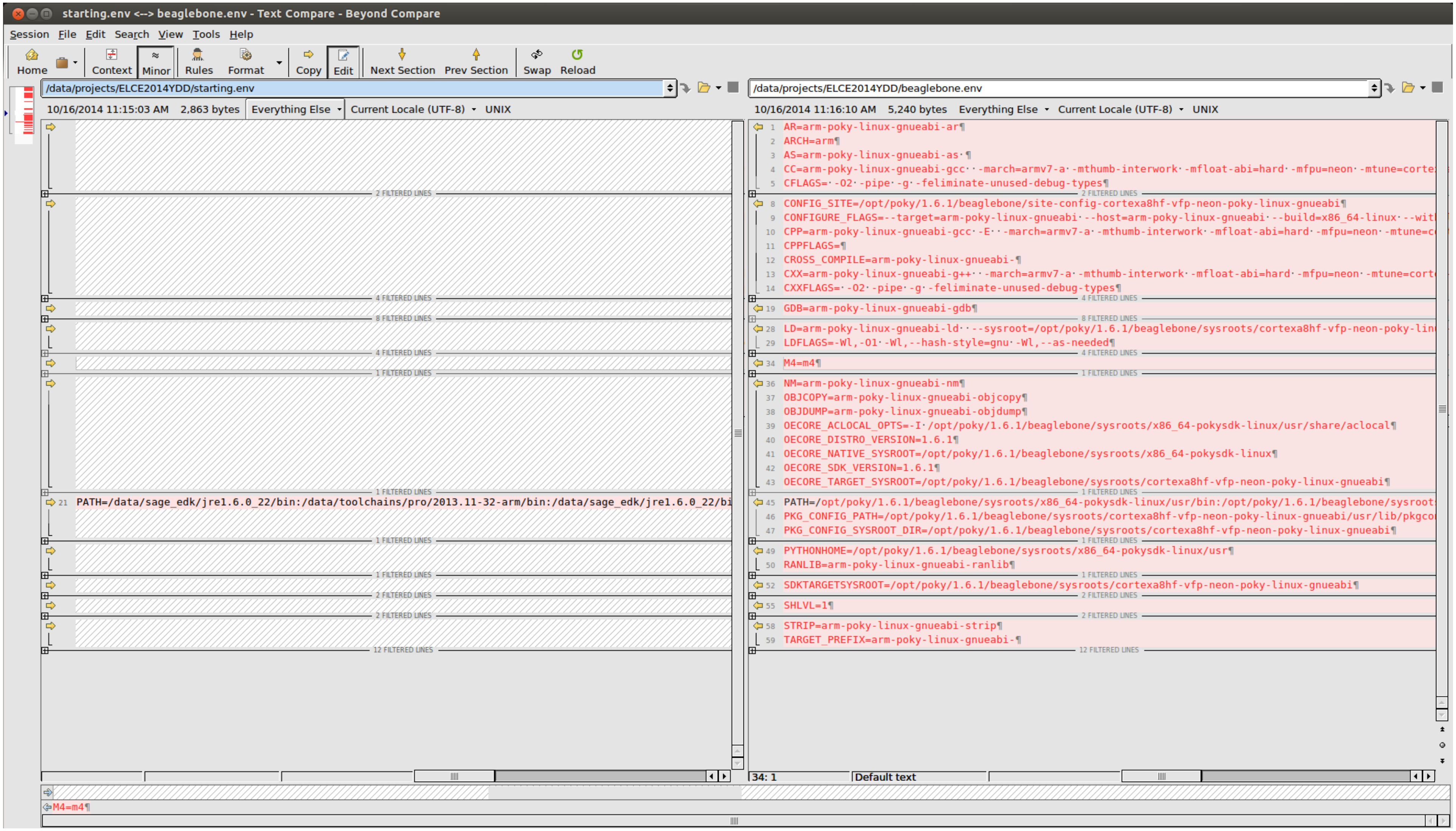
Using an SDK

- Execute the SDK environment script



```
shudson@ronin:[3]:~$ . /opt/poky/1.6.1/beaglebone/environment-setup-cortexa8hf-vfp-neon-poky-linux-gnueabi
shudson@ronin:[3]:~$ which gcc
/usr/bin/gcc
shudson@ronin:[3]:~$ gcc --version
gcc (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

shudson@ronin:[3]:~$ 
[0] 0:bash*                                                                      "ronin" 03:41 16-Oct-14
```

Let’s look at the shell environment


starting.env <--> beaglebone.env - Text Compare - Beyond Compare
Session File Edit Search View Tools Help
Home
Context
Minor
Rules
Format
Copy
Edit
Next Section
Prev Section
Swap
Reload

/data/projects/ELCE2014YDD/starting.env

10/16/2014 11:15:03 AM 2,863 bytes Everything Else Current Locale (UTF-8) UNIX

```
2 FILTERED LINES
4 FILTERED LINES
8 FILTERED LINES
4 FILTERED LINES
1 FILTERED LINES
1 FILTERED LINES
PATH=/data/sage_edk/jre1.6.0_22/bin:/data/toolchains/pro/2013.11-32-arm/bin:/data/sage_edk/jre1.6.0_22/bi
1 FILTERED LINES
1 FILTERED LINES
2 FILTERED LINES
2 FILTERED LINES
12 FILTERED LINES
```

/data/projects/ELCE2014YDD/beaglebone.env

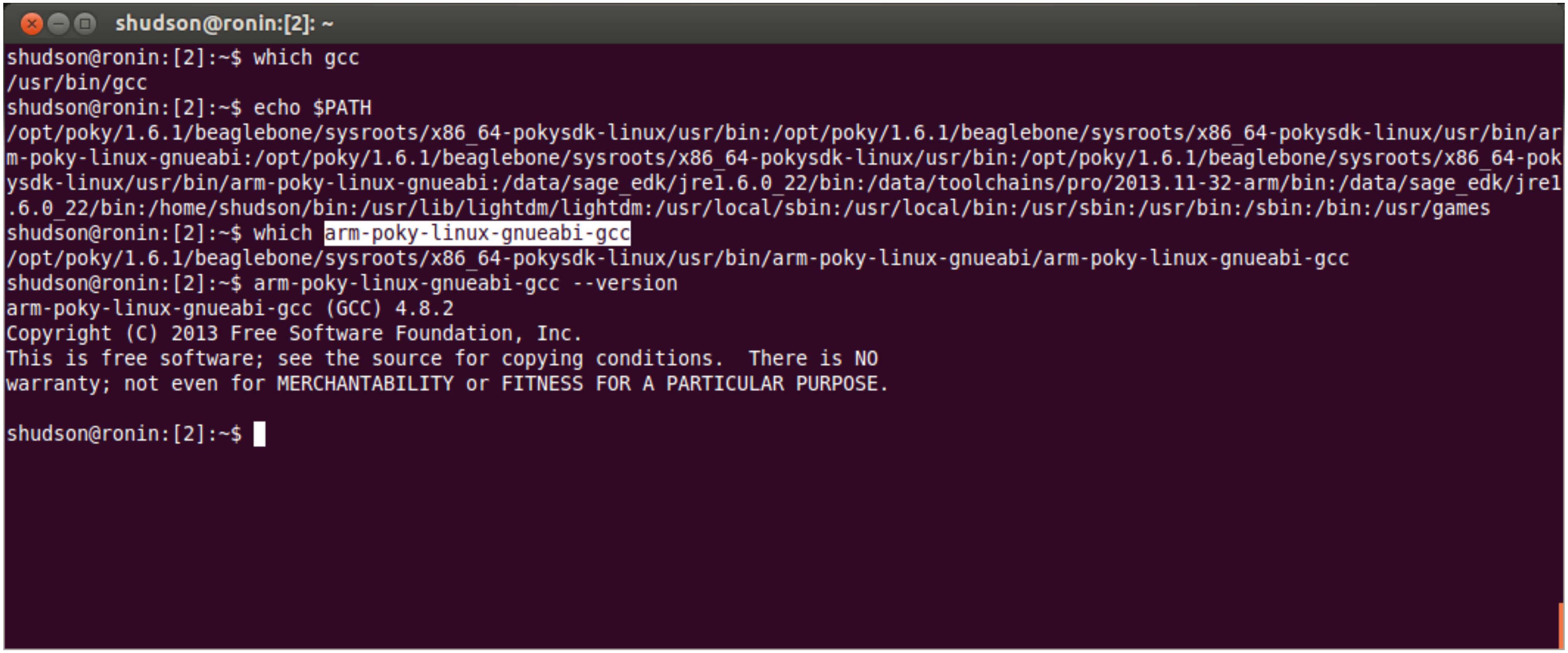
10/16/2014 11:16:10 AM 5,240 bytes Everything Else Current Locale (UTF-8) UNIX

```
AR=arm-poky-linux-gnueabi-ar¶
ARCH=arm¶
AS=arm-poky-linux-gnueabi-as·¶
CC=arm-poky-linux-gnueabi-gcc··-march=armv7-a·-mthumb-interwork·-mfloat-abi=hard·-mfpu=neon·-mtune=corte
CFLAGS=·-O2·-pipe·-g·-feliminate-unused-debug-types¶
2 FILTERED LINES
CONFIG_SITE=/opt/poky/1.6.1/beaglebone/site-config-cortexa8hf-vfp-neon-poky-linux-gnueabi¶
CONFIGURE_FLAGS=--target=arm-poky-linux-gnueabi·--host=arm-poky-linux-gnueabi·--build=x86_64-linux·--wit
CPP=arm-poky-linux-gnueabi-gcc·-E··-march=armv7-a·-mthumb-interwork·-mfloat-abi=hard·-mfpu=neon·-mtune=c
CPPFLAGS=¶
CROSS_COMPILE=arm-poky-linux-gnueabi-¶
CXX=arm-poky-linux-gnueabi-g++··-march=armv7-a·-mthumb-interwork·-mfloat-abi=hard·-mfpu=neon·-mtune=cort
CXXFLAGS=·-O2·-pipe·-g·-feliminate-unused-debug-types¶
4 FILTERED LINES
GDB=arm-poky-linux-gnueabi-gdb¶
8 FILTERED LINES
LD=arm-poky-linux-gnueabi-ld··--sysroot=/opt/poky/1.6.1/beaglebone/sysroots/cortexa8hf-vfp-neon-poky-lin
LDFLAGS=-Wl,-O1·-Wl,--hash-style=gnu·-Wl,--as-needed¶
4 FILTERED LINES
M4=m4¶
1 FILTERED LINES
NM=arm-poky-linux-gnueabi-nm¶
OBJCOPY=arm-poky-linux-gnueabi-objcopy¶
OBJDUMP=arm-poky-linux-gnueabi-objdump¶
OECORE_ACLOCAL_OPTS=-I·/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/share/aclocal¶
OECORE_DISTRO_VERSION=1.6.1¶
OECORE_NATIVE_SYSROOT=/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux¶
OECORE_SDK_VERSION=1.6.1¶
OECORE_TARGET_SYSROOT=/opt/poky/1.6.1/beaglebone/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi¶
1 FILTERED LINES
PATH=/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/bin:/opt/poky/1.6.1/beaglebone/sysroot
PKG_CONFIG_PATH=/opt/poky/1.6.1/beaglebone/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi/usr/lib/pkgco
PKG_CONFIG_SYSROOT_DIR=/opt/poky/1.6.1/beaglebone/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi¶
1 FILTERED LINES
PYTHONHOME=/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr¶
RANLIB=arm-poky-linux-gnueabi-ranlib¶
1 FILTERED LINES
SDKTARGETSYSROOT=/opt/poky/1.6.1/beaglebone/sysroots/cortexa8hf-vfp-neon-poky-linux-gnueabi¶
2 FILTERED LINES
SHLVL=1¶
2 FILTERED LINES
STRIP=arm-poky-linux-gnueabi-strip¶
TARGET_PREFIX=arm-poky-linux-gnueabi-¶
12 FILTERED LINES
```

34: 1 Default text

```
M4=m4¶
```

Let’s look at the path

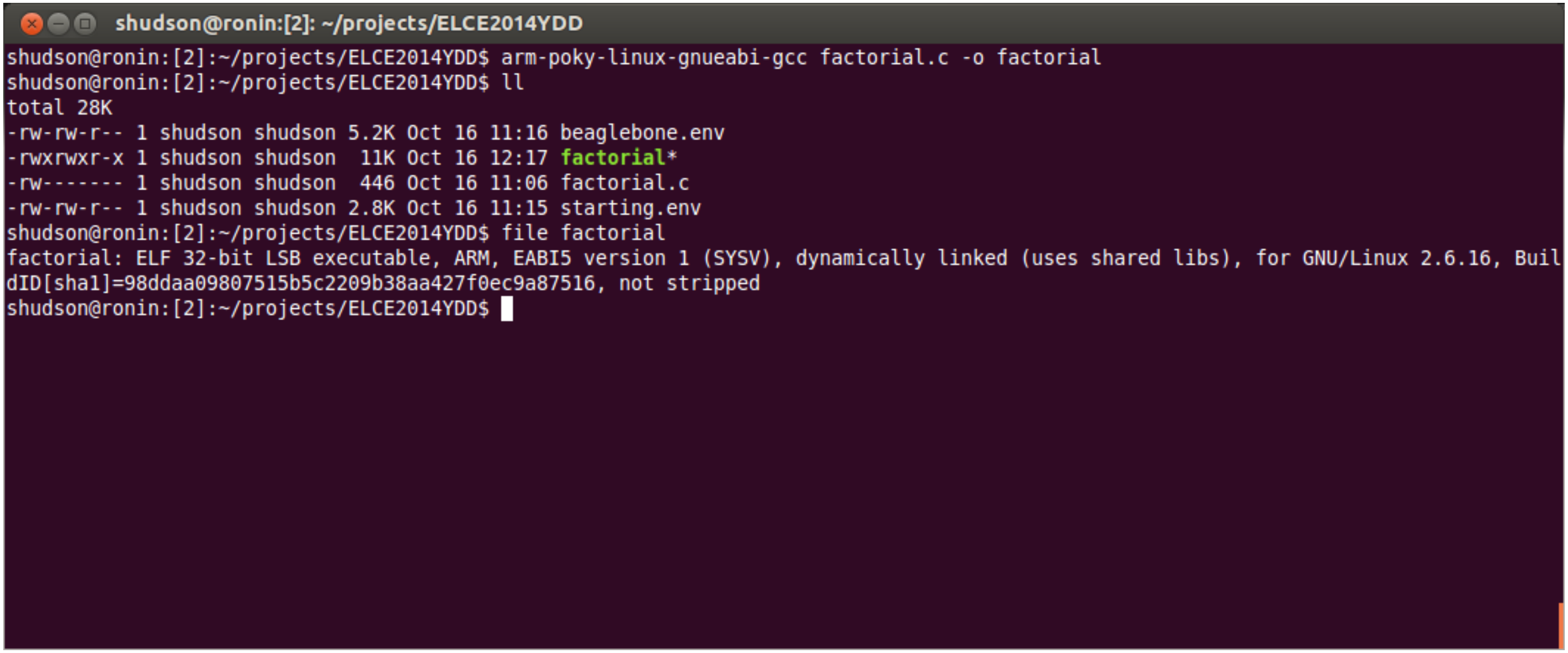


```
shudson@ronin:[2]:~$ which gcc
/usr/bin/gcc
shudson@ronin:[2]:~$ echo $PATH
/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/bin:/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/bin/ar
m-poky-linux-gnueabi:/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/bin:/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pok
ysdk-linux/usr/bin/arm-poky-linux-gnueabi:/data/sage_edk/jre1.6.0_22/bin:/data/toolchains/pro/2013.11-32-arm/bin:/data/sage_edk/jre1
.6.0_22/bin:/home/shudson/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
shudson@ronin:[2]:~$ which arm-poky-linux-gnueabi-gcc
/opt/poky/1.6.1/beaglebone/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gcc
shudson@ronin:[2]:~$ arm-poky-linux-gnueabi-gcc --version
arm-poky-linux-gnueabi-gcc (GCC) 4.8.2
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

shudson@ronin:[2]:~$ 
```

A trivial code example: factorial.c

```
#include <stdio.h>

int main()
{
      int c, n, fact = 1;

        printf("Enter a number to calculate it's factorial\n");
          scanf("%d", &n);

            for (c = 1; c <= n; c++)
                    fact = fact * c;

              printf("Factorial of %d = %d\n", n, fact);

                return 0;
}
```

Build the example



shudson@ronin:[2]: ~/projects/ELCE2014YDD

```
shudson@ronin:[2]:~/projects/ELCE2014YDD$ arm-poky-linux-gnueabi-gcc factorial.c -o factorial
shudson@ronin:[2]:~/projects/ELCE2014YDD$ ll
total 28K
-rw-rw-r-- 1 shudson shudson 5.2K Oct 16 11:16 beaglebone.env
-rwxrwxr-x 1 shudson shudson  11K Oct 16 12:17 factorial*
-rw------- 1 shudson shudson  446 Oct 16 11:06 factorial.c
-rw-rw-r-- 1 shudson shudson 2.8K Oct 16 11:15 starting.env
shudson@ronin:[2]:~/projects/ELCE2014YDD$ file factorial
factorial: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.16, Buil
dID[sha1]=98ddaa09807515b5c2209b38aa427f0ec9a87516, not stripped
shudson@ronin:[2]:~/projects/ELCE2014YDD$ 
```

A Sample Application Dev Workflow

- Receive SDK from platform team
- Extract it to local drive
- Develop and debug application
  - Use the SDK wrapper script to wrap the application build
  - iterate
- When the application is ready to be added to the platform, the application developer either creates a recipe, or asks the platform team to create a recipe to add it to the SDK
- Repeat, as necessary

Final Thoughts

- Most of the heavy lifting has already been done for you in generating and using the SDKs
- Define workflows that work for your situation
- Same process applies and works when you add the meta-qt5 layer (see Denys Dmytriyenko's presentation)
- Some improvements are being discussed already:
  - documenting best practices for workflows
  - enhancing the tooling around workflows

QUESTIONS?

REFERENCES

References

- Yocto Project Application Developer's Guide v. 1.6.1
  - Refer to section: 3.4 - Optionally Building a Toolchain Installer
- Yocto Project Reference v. 1.6.1
  - Refer to section: 3.5.6 - SDK Generation