2nd exercise

NXSOL-OJT-2024

Exported on 02/27/2024

Table of Contents

1 1:

1.1 Set up the environment

1.1.1 Download base poky

```
git clone -b kirkstone git://git.yoctoproject.org/poky.git
```

1.1.2 Set the build environment

```
source poky/oe-init-build-env ; cd ..
```

1.2 Create a custom layer

1.2.1 Create a layer

meta-[name] (e.g. meta-qwe)

```
bitbake-layers create-layer meta-[name]
```

1.2.2 Add the layer

Add dependency layers in `build/conf/bblayers.conf`

Open

bblayers.conf
~/me/yocto/build/conf

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
  ${TOPDIR}/../poky/meta \
  ${TOPDIR}/../poky/meta-poky \
  ${TOPDIR}/../poky/meta-yocto-bsp \
  ${TOPDIR}/../meta-qwe \
  ${TOPDIR}/../meta-tensorflow \
  ${TOPDIR}/../meta-openembedded/meta-python \
  ${TOPDIR}/../meta-openembedded/meta-oe \
  "
```

1.3 Create a core image recipe

1.3.1 Create an "images" directory in the layer created above

```
mkdir -p [root]/meta-[name]/recipes-core/images
```

1.3.2 Create [name]-image.bb in this directory and write the image recipe in it

```
DESCRIPTION = "A core image for QWE"
LICENSE = "MIT"

# Core files for basic console boot
IMAGE_INSTALL = "packagegroup-core-boot"

# Add our desired extra files
IMAGE_INSTALL += "psplash dropbear"

inherit core-image

IMAGE_ROOTFS_SIZE ?= "8192"
```

1.4 Create a hello world package

1.4.1 Create hello world source

```
mkdir -p [root]/meta-[name]/recipes-core/hello/files
touch [root]/meta-[name]/recipes-core/hello/files/hello.c
```

hello.c

```c
#include <stdio.h>

int main(int argc, char **argv) {
        printf("Hello World\n");
        return 0;
}
```

1.4.2 Create a hello world recipe

```
touch [root]/meta-[name]/recipes-core/hello/hello_1.0.bb
```

```
DESCRIPTION = "Hello World example"
LICENSE = "MIT"

LIC_FILES_CHKSUM = "file://${COREBASE}/meta/
COPYING.MIT;md5=3da9cfbcb788c80a0384361b4de20420"

S = "${WORKDIR}"

SRC_URI = "file://hello.c"

do_compile() {
        ${CC} ${CFLAGS} ${LDFLAGS} hello.c -o hello
}

do_install() {
        install -d -m 0755 ${D}/${bindir}
        install -m 0755 hello ${D}/${bindir}/hello
}
```

1.5 Add the hello world package to the core image recipe

`[root]/meta-[name]/recipes-core/images/[name]-image.bb`

Open qwe-image.bb ~/Downloads/me/Yocto_Project_seminar/assignment1/meta-qwe/recipes-core/images Save

```
DESCRIPTION = "A core image for QWE"
LICENSE = "MIT"

# Core files for basic console boot
IMAGE_INSTALL = "packagegroup-core-boot"

# Add our desired extra files
IMAGE_INSTALL += "psplash dropbear hello"

inherit core-image

IMAGE_ROOTFS_SIZE ?= "8192"
```

1.6 Build and run the custom image

```
bitbake -k qwe-image
runqemu qwe-image
```

```
[    1.319190] ata1: SATA link down (SStatus 0 SControl 300)
[    1.321032] ata3: SATA link up 1.5 Gbps (SStatus 113 SControl 300)
[    1.323536] ata3.00: ATAPI: QEMU DVD-ROM, 2.5+, max UDMA/100
[    1.324760] ata3.00: applying bridge limits
[    1.326629] ata3.00: configured for UDMA/100
[    1.331712] scsi 2:0:0:0: CD-ROM            QEMU     QEMU DVD-ROM     2.5+ PQ: 0 ANSI: 5
[    1.357466] sr 2:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    1.358805] cdrom: Uniform CD-ROM driver Revision: 3.20
[    1.450961] input: QEMU QEMU USB Tablet as /devices/pci0000:00/0000:00:1d.7/usb1/1-1/1-1:1.0/0003:0627:0001.0001/input/input4
[    1.458719] hid-generic 0003:0627:0001.0001: input: USB HID v0.01 Mouse [QEMU QEMU USB Tablet] on usb-0000:00:1d.7-1/input0
[    1.536654] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042/serio1/input/input3
[    1.562346] IP-Config: Complete:
[    1.567210]      device=eth0, hwaddr=52:54:00:12:34:02, ipaddr=192.168.7.2, mask=255.255.255.0, gw=192.168.7.1
[    1.568938]      host=192.168.7.2, domain=, nis-domain=(none)
[    1.570776]      bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
[    1.570805]      nameserver0=8.8.8.8
[    1.578422] md: Waiting for all devices to be available before autodetect
[    1.579673] md: If you don't use raid, use raid=noautodetect
[    1.580991] md: Autodetecting RAID arrays.
[    1.583334] md: autorun ...
[    1.584559] md: ... autorun DONE.
[    1.610280] EXT4-fs (vda): recovery complete
[    1.618208] EXT4-fs (vda): mounted filesystem with ordered data mode. Opts: (null). Quota mode: disabled.
[    1.623419] VFS: Mounted root (ext4 filesystem) on device 253:0.
[    1.626043] devtmpfs: mounted
[    1.711304] tsc: Refined TSC clocksource calibration: 1996.756 MHz
[    1.712824] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x39906c43571, max_idle_ns: 881590820589 ns
[    1.714376] clocksource: Switched to clocksource tsc
[    1.774445] Freeing unused kernel image (initmem) memory: 1880K
[    1.779824] Write protecting the kernel read-only data: 22528k
[    1.787203] Freeing unused kernel image (text/rodata gap) memory: 2036K
[    1.793601] Freeing unused kernel image (rodata/data gap) memory: 484K
[    1.795978] Run /sbin/init as init process
INIT: version 3.01 booting
[    2.031367] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
warning: FBIOPUT_VSCREENINFO failed, double buffering disabled
Starting udev
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
Starting Dropbear SSH server: dropbear.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 4.0.15 qemux86-64 /dev/tty1

qemux86-64 login: root
root@qemux86-64:~# hello
Hello World
root@qemux86-64:~#
```

2 2:

2.1 런타임 패키지 변수에 대해 설명하시오:

- `RRECOMMENDS`:
  - **Description**: This variable lists packages that are not essential but are recommended to be installed along with the package being defined. If these recommended packages are available, they will be installed by default.
  - **Usage**: It's used when a package can provide additional functionality if another package is present, but can still function without it.
- `RSUGGESTS`:
  - **Description**: Similar to `RRECOMMENDS`, `RSUGGESTS` lists packages that are suggested to be used with the package being defined, but are even less critical. These suggestions are typically not installed by default and are just informational.
  - **Usage**: It's useful for indicating optional packages that could enhance the functionality or user experience but are not directly tied to the package's operation.
- `RPROVIDES`:
  - **Description**: This variable is used to specify that the package being defined provides the features or functionality of another package (or multiple packages). It's a way of saying, "Installing this package is as good as installing the listed ones."
  - **Usage**: Commonly used in situations where multiple packages can fulfill the same dependency, or when creating a virtual/meta-package that represents a group of packages.
- `RCONFLICTS`:
  - **Description**: `RCONFLICTS` is used to declare that the package being defined cannot be installed at the same time as the listed package(s). It's a way to prevent incompatible packages from being installed together.
  - **Usage**: Important for ensuring that packages that would cause problems if installed together are kept separate. For example, two packages that try to install the same file.
- `RREPLACES`:
  - **Description**: This variable is used to indicate that the package being defined should replace the listed packages. It is often used in conjunction with `RCONFLICTS` to specify that this package not only conflicts with but also supersedes another package.
  - **Usage**: Commonly used during upgrades or when one package is meant to entirely substitute another, effectively making the old package obsolete.