Using Custom Users/Groups in Yocto 1.1

Scott Garman <scott.a.garman@intel.com>

Why do we need this?

- Previously, custom users/groups could only be created in postinstall scripts run at image first boot
- There was no framework to ensure that these user/group additions were not in conflict with other packages
- As a result, almost no recipes made use of this, so nearly everything in our rootfs images were owned by root:root

Why do we need this?

- System daemons in particular make heavy use of custom users/groups in order to enforce privilege separation – a fundamental security model
- The lack of this feature has been a significant flaw with Yocto until now

Abstract Requirements

- A recipe needs to be able to define:
  - One or more custom user names
  - One or more custom group names
  - Associate any/all files in a package with the above users/groups
  - Allow for full control over the useradd/groupadd commands (e.g, specify home directory and other options)
  - Cannot require builds to be run with root privileges (i.e, make appropriate use of pseudo)

Implementation Notes

- Logic is encapsulated in **useradd.bbclass**
- A custom /etc/passwd and /etc/group are maintained in the **target sysroot** during builds
- Pseudo has a PSEUDO_PASSWD environment variable which is pointed to the target sysroot
- The *useradd* and *groupadd* utilities from *shadow-native* are run under pseudo when generating images
- Package **preinstall** scripts are generated which run when a package is manually installed on the target

How to Use It

- Example recipe can be found in **meta-skeleton/recipes-skeleton/useradd/**
- **inherit useradd**
- Specify output packages which include custom users/groups in the **USERADD_PACKAGES** variable

How to Use It

- **USERADD_PARAM_${PN}** is set to the command-line options to the **useradd** command
- **GROUPADD_PARAM_${PN}** is set to the command-line options to the **groupadd** command
- Separate multiple useradd/groupadd command options with a semicolon
- Use chown/chgrp commands in **do_install()** or **do_install_append()**

useradd-example.bb

```
inherit useradd

USERADD_PACKAGES = "${PN} ${PN}-user3“

USERADD_PARAM_${PN} = “-d /home/user1 -r -s /bin/bash user1; -d
   /home/user2 -r -s /bin/bash user2"

# user3 will be managed in the useradd-example-user3 pkg:
USERADD_PARAM_${PN}-user3 = “-d /home/user3 -r -s /bin/bash
   user3"

GROUPADD_PARAM_${PN} = “group1; group2"

# Likewise, we'll manage group3 in the useradd-example-user3 pkg:
GROUPADD_PARAM_${PN}-user3 = “group3"
```

useradd-example.bb (cont)

```
do_install () {
    # The new users and groups are created before the do_install
    # step, so you are now free to make use of them:
    chown -R user1:group1 ${D}/usr/share/user1

    chown -R user2 ${D}/usr/share/user2
    chown -R user3 ${D}/usr/share/user3

    chgrp -R group2 ${D}/usr/share/user2
    chgrp -R group3 ${D}/usr/share/user3
}

FILES_${PN} = "/usr/share/user1/* /usr/share/user2/*"
FILES_${PN}-user3 = "/usr/share/user3/*"
```

Tip: Viewing Pre/Post Install Scripts from a Package

- If you'd like to examine the pre- or post-install scripts from a package without installing it:
- RPM:
  - **rpm -qp --scripts package.rpm**
- IPK/DEB:
  - **ar -vx package.ipk**
  - **tar xvf control.tar.gz**

Q&A

Questions?