Linux Yocto 3.5

Tworzenie Daemona, Ustawienie statycznego IP

Autor	Bartłomiej Krasoń
Wersja	1
Data modyfikacji	03-01-2019

Spis treści

Wprowadzenie

Niniejszy dokument opisuje jak stworzyć program działający “w tle” na systemie Linux (dystrybucja Yocto 3.5). Program taki nazywa się serwisem, bądź dla systemu Linux stosuje się osobliwą nazwę *daemon*. Każdy serwis w systemie może być skonfigurowany szczegółowo pod siebie, co też zostanie opisane w tym dokumencie.

Utworzenie programu

Każdy serwis jest tak de facto programem, dlatego proces tworzenia serwisu rozpoczynamy od napisania programu. Zazwyczaj serwis jest programem który działa w nieskończonej pętli, co oznacza tyle, że działa cały czas “w tle” działania systemu. Każdy system po zainstalowaniu na komputerze posiada już sporą część serwisów systemowych, które odpowiadają za zadania typu: przekazywania powiadomień użytkownikowi, logowanie różnych użytkowników do systemu, działanie schowka itp.. Nasz serwis będzie odpowiedzialny za pobieranie oraz przesyłanie wiadomości przy wykorzystaniu protokołu MQTT, dlatego chcemy aby działał cały czas “w tle” za każdym razem po uruchomieniu systemu. Program stanowiący serwis może być napisany w każdym języku. Jedynym wymaganiem co do programu jest, że musimy on być w stanie uruchomiony z linii poleceń. Przykładowo dla skryptu w Pythonie, będziemy potrzebowali jego interpretera zainstalowanego w systemie.

Przykładowy kod programy używany w tym przykładzie:

```
#!/usr/bin/python
import paho.mqtt.client as mqtt
file = open("data.txt", "a")
def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))
   print("Hello, you successfully connected to your local server!")
   client.subscribe("test/#")
def on_message(client, userdata, msg):
     print("Writing data... => " + msg.topic + " " + str(msg.payload))
   file.write("Topic: " + msg.topic + ", Message: " + msg.payload);

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_forever()
```

Jak widzimy, przykładowy program działa w nieskończonej pętli.

Dodanie serwisu

Utworzenie pliku serwisu

Aby dodać serwis w systemie Linux Yocto 3.5 należy w pierwszy kroku utworzyć odpowiedni plik stanowiący konfigurację serwisu i umożliwiający uruchomienie go w systemie. Plik ten ma rozszerzenie **`*.service`** i powinien znajdować się w katalogu: **`/lib/systemd/system/`**.

Zawartość pliku **`LocalSub.service`** dla naszego przykładu:

```
[Unit]
Description=LocalSub
DefaultDependencies=no
Requires=network-online.target
After=network.target network-online.target

[Service]
ExecStart=/home/root/LocalSub.py
WorkingDirectory=/home/root
Restart=on-failure
RestartSec=30s

[Install]
WantedBy=multi-user.target
```

Plik podzielony jest na sekcje. W naszym przykładzie:

- sekcja **[Unit]** - ogólna konfiguracja serwisu, gdzie możemy ustalić m. in. jego nazwę (Description) oraz moment w rozruchu systemu, kiedy ma zostać uruchomiony dany serwis (Requires, After)
- sekcja **[Service]** - szczegółowa konfiguracja serwisu, podajemy ścieżkę wywołania programu (ExecStart) itp. oraz dane m. in. dotyczące kiedy i jak często serwis ma się restartować.
- sekcja **[Install]** - określa miejsce wpisu tego serwisu w danej lokalizacji **`*.wants`** – która określa moment, w jakim system zainstaluje dany serwis. W naszym przykładzie jest to lokalizacja:
  `/etc/systemd/system/multi-user.target.wants/`

  Aby sprawdzić konfigurację miejsca instalacji wszystkich serwisów można skorzystać z komend:

  ```
  $ ls /etc/systemd/system/*.wants/
  ```

  lub

  ```
  $ systemctl list-unit-files --type=service
  ```

Zamontowanie serwisu w systemie

Poniższa lista kroków opisuje jak zamontować przygotowany uprzednio serwis **`LocalSub.service`** w systemie:

1. Upewnienie się że plik serwisu istnieje w lokalizacji:

   `lib/systemd/system/LocalSub.service`

2. W konsoli montujemy istniejący serwis:

   ```
   $ systemctl enable LocalSub.service
   ```

3. Przeładowujemy listę serwisów:

   ```
   $ systemctl daemon-reload
   ```

4. Można sprawdzić status serwisu:

   ```
   $ systemctl status LocalSub.service
   ```

5. Można zrestartować serwis w każdym momencie:

   ```
   $ systemctl restart LocalSub.service
   ```

6. Aby sprawdzić czy serwis działa w tle można wylistować wszystkie procesy poleceniem:

   ```
   $ ps
   ```

Ustawienie statycznego IP

Aby w stystemie Linux Yocto 3.5 zagwarantować ustawienie statycznego ip, należy edytować plik:

`/etc/wpa_supplicant/wpa_cli-actions.sh`

Plik wpa_cli-actions.sh przed edycją

```
#!/bin/sh
#
# This script file is passed as parameter to wpa_cli, started as a
daemon,
# so that the wpa_supplicant events are sent to this script
# and actions executed, like :
#    - start DHCP client when STA is connected.
#    - stop DHCP client when STA is disconnected.
#    - start DHCP client when P2P-GC is connected.
#    - stop DHCP server when  P2P-GO is disconnected.
#
# This script skips events if connmand (connman.service) is started
# Indeed, it is considered that the Wifi connection is managed through
# connmand and not wpa_cli
#

IFNAME=$1
```

```
CMD=$2

kill_daemon() {
    NAME=$1
    PF=$2

    if [ ! -r $PF ]; then
        return
    fi

    PID=`cat $PF`
    if [ $PID -gt 0 ]; then
        if ps | grep $NAME | grep $PID; then
            kill $PID
        fi
    fi
    if [ -r $PF ]; then
        # file can be removed by the deamon when killed
        rm $PF
    fi
}

echo "event $CMD received from wpa_supplicant"

# if Connman is started, ignore wpa_supplicant
# STA connection event because the DHCP connection
# is triggerd by Connman
if [ `systemctl is-active connman` == "active" ] ; then
    if [ "$CMD" = "CONNECTED" ] || [ "$CMD" = "DISCONNECTED" ] ; then
        echo "event $CMD ignored because Connman is started"
        exit 0
    fi
fi

if [ "$CMD" = "CONNECTED" ]; then
    kill_daemon udhcpc /var/run/udhcpc-$IFNAME.pid
    udhcpc -i $IFNAME -p /var/run/udhcpc-$IFNAME.pid -S
fi

if [ "$CMD" = "DISCONNECTED" ]; then
    kill_daemon udhcpc /var/run/udhcpc-$IFNAME.pid
    ifconfig $IFNAME 0.0.0.0
fi

if [ "$CMD" = "P2P-GROUP-STARTED" ]; then
    GIFNAME=$3
    if [ "$4" = "GO" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        ifconfig $GIFNAME 192.168.42.1 up
        cp /etc/wpa_supplicant/udhcpd-p2p.conf
/etc/wpa_supplicant/udhcpd-p2p-itf.conf
        sed -i "s/INTERFACE/$GIFNAME/" /etc/wpa_supplicant/udhcpd-p2p-
itf.conf
        udhcpd /etc/wpa_supplicant/udhcpd-p2p-itf.conf
    fi
    if [ "$4" = "client" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        kill_daemon udhcpd /var/run/udhcpd-$GIFNAME.pid
```

```
        udhcpc -i $GIFNAME -p /var/run/udhcpc-$GIFNAME.pid
    fi
fi

if [ "$CMD" = "P2P-GROUP-REMOVED" ]; then
    GIFNAME=$3
    if [ "$4" = "GO" ]; then
        kill_daemon udhcpd /var/run/udhcpd-$GIFNAME.pid
        ifconfig $GIFNAME 0.0.0.0
    fi
    if [ "$4" = "client" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        ifconfig $GIFNAME 0.0.0.0
    fi
fi
```

Plik wpa_cli-actions.sh po edycji

Zmianie ulega fragment zaznaczony na żółto. Fragmenty pogrubione są do ustawienia przez programistę.

```
#!/bin/sh
#
# This script file is passed as parameter to wpa_cli, started as a
daemon,
# so that the wpa_supplicant events are sent to this script
# and actions executed, like :
#    - start DHCP client when STA is connected.
#    - stop DHCP client when STA is disconnected.
#    - start DHCP client when P2P-GC is connected.
#    - stop DHCP server when  P2P-GO is disconnected.
#
# This script skips events if connmand (connman.service) is started
# Indeed, it is considered that the Wifi connection is managed through
# connmand and not wpa_cli
#

IFNAME=$1
CMD=$2

kill_daemon() {
    NAME=$1
    PF=$2

    if [ ! -r $PF ]; then
        return
    fi

    PID=`cat $PF`
    if [ $PID -gt 0 ]; then
        if ps | grep $NAME | grep $PID; then
            kill $PID
        fi
    fi
    if [ -r $PF ]; then
        # file can be removed by the deamon when killed
        rm $PF
```

```
    fi
}

echo "event $CMD received from wpa_supplicant"

# if Connman is started, ignore wpa_supplicant
# STA connection event because the DHCP connection
# is triggerd by Connman
if [ `systemctl is-active connman` == "active" ] ; then
    if [ "$CMD" = "CONNECTED" ] || [ "$CMD" = "DISCONNECTED" ] ; then
        echo "event $CMD ignored because Connman is started"
        exit 0
    fi
fi

if [ "$CMD" = "CONNECTED" ]; then
#    kill_daemon udhcpc /var/run/udhcpc-$IFNAME.pid
#    udhcpc -i $IFNAME -p /var/run/udhcpc-$IFNAME.pid -S
     ifconfig $IFNAME 192.168.1.200 netmask 255.255.255.0
     route add default gw funbox.home
fi

if [ "$CMD" = "DISCONNECTED" ]; then
    kill_daemon udhcpc /var/run/udhcpc-$IFNAME.pid
    ifconfig $IFNAME 0.0.0.0
fi

if [ "$CMD" = "P2P-GROUP-STARTED" ]; then
    GIFNAME=$3
    if [ "$4" = "GO" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        ifconfig $GIFNAME 192.168.42.1 up
        cp /etc/wpa_supplicant/udhcpd-p2p.conf
/etc/wpa_supplicant/udhcpd-p2p-itf.conf
        sed -i "s/INTERFACE/$GIFNAME/" /etc/wpa_supplicant/udhcpd-p2p-
itf.conf
        udhcpd /etc/wpa_supplicant/udhcpd-p2p-itf.conf
    fi
    if [ "$4" = "client" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        kill_daemon udhcpd /var/run/udhcpd-$GIFNAME.pid
        udhcpc -i $GIFNAME -p /var/run/udhcpc-$GIFNAME.pid
    fi
fi

if [ "$CMD" = "P2P-GROUP-REMOVED" ]; then
    GIFNAME=$3
    if [ "$4" = "GO" ]; then
        kill_daemon udhcpd /var/run/udhcpd-$GIFNAME.pid
        ifconfig $GIFNAME 0.0.0.0
    fi
    if [ "$4" = "client" ]; then
        kill_daemon udhcpc /var/run/udhcpc-$GIFNAME.pid
        ifconfig $GIFNAME 0.0.0.0
    fi
fi
```