目次

カスタムアプリケーションの自動起動設定

作成したアプリケーションを、MA-E3xxの電源をONしたときに自動で起動させる設定の方法です。

古典的な方法 (rc.local)

一般的な Linux のディストリビューション同様、Ubuntu Linux にも rc.local ファイルが用意されています。

```
user1@plum:~$ cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

exit 0
user1@plum:~$
```

コメントの通り、マルチユーザランレベルの最後に実行されます。

```
user1@plum:~$ ls -l /etc/rc2.d/
total 4
-rw-r--r-- 1 root root 677 Mar 13 10:42 README
lrwxrwxrwx 1 root root  17 Mar  6 09:55 S15dnsmasq -> ../init.d/dnsmasq
lrwxrwxrwx 1 root root  22 Feb 18 13:53 S19cpufrequtils ->
../init.d/cpufrequtils
lrwxrwxrwx 1 root root  18 May 28 14:43 S20ddclient -> ../init.d/ddclient
lrwxrwxrwx 1 root root  15 Feb 17 14:44 S20nginx -> ../init.d/nginx
lrwxrwxrwx 1 root root  13 Feb 18 18:07 S23ntp -> ../init.d/ntp
lrwxrwxrwx 1 root root  19 Feb 18 17:53 S70dns-clean -> ../init.d/dns-clean
lrwxrwxrwx 1 root root  18 Feb 18 17:53 S70pppd-dns -> ../init.d/pppd-dns
lrwxrwxrwx 1 root root  14 Mar 14 12:39 S75sudo -> ../init.d/sudo
lrwxrwxrwx 1 root root  18 Feb 17 12:58 S99rc.local -> ../init.d/rc.local
<---- このシンボリックリンク
user1@plum:~$
```

このファイルに作成したアプリケーションを起動する処理を記述することで実現できます。

UpstartのJobとして定義する方法

前述の rc.local による起動の場合、単に実行するだけですので下記機能がありません。

- アプリケーションの死活監視 (停止してしまった場合の再起動など)
- stop/restart/reload 等のジョブ制御

Upstart の Job にすると、上記が実現可能となります。

既存の Job

/etc/init 以下に、Upstart の Job ファイルが配置されています。

```
user1@plum:~$ ls /etc/init
bootmisc.sh.conf             mobile_watch.conf                rc.conf
butterfly.conf               mountall-bootclean.sh.conf       rcS.conf
checkfs.sh.conf              mountall-net.conf                rpcbind-
boot.conf
checkroot-bootclean.sh.conf  mountall-reboot.conf             rpcbind.conf
checkroot.sh.conf            mountall-shell.conf              rsyslog.conf
console-detect.conf          mountall.conf                    screen-
cleanup.conf
console.conf                 mountall.sh.conf                 shutdown-
led.conf
container-detect.conf        mountdevsubfs.sh.conf            shutdown.conf
control-alt-delete.conf      mounted-debugfs.conf             ssh.conf
cron.conf                    mounted-dev.conf                 statd-
mounting.conf
dbus.conf                    mounted-run.conf                 statd.conf
disable-usb-wakeup.conf      mounted-tmp.conf                 ttyO0.conf
dmesg.conf                   mounted-var.conf                 ttyO1.conf
export_di.conf               mountkernfs.sh.conf              udev-
finish.conf
flush-early-job-log.conf     mountnfs-bootclean.sh.conf       udev.conf
gssd-mounting.conf           mountnfs.sh.conf
udevmonitor.conf
gssd.conf                    mtab.sh.conf
udevtrigger.conf
hostname.conf                network-interface-container.conf  ufw.conf
hwclock-save.conf            network-interface-security.conf   upstart-file-
bridge.conf
hwclock.conf                 network-interface.conf            upstart-
```

```
socket-bridge.conf
idmapd-mounting.conf          networking.conf                  upstart-udev-
bridge.conf
idmapd.conf                   passwd.conf                      usb-
modeswitch-upstart.conf
initsw.conf                   portmap-wait.conf                wait-for-
state.conf
kmod.conf                     procps.conf                      zabbix-
agent.conf
mgetty.conf                   rc-sysinit.conf                  zram-
config.conf
user1@plum:~$
```

MA-E3xx 用に追加・カスタムしたものは下記になります。

No.	Filename	Info
1	butterfly.conf	Web Terminal “Butterfly”[1]
2	console-detect.conf	Consoleとして定義されているポート検出
3	export_di.conf	DI(Digital IN) GPIO を /tmp/di 以下に export
4	initsw.conf	InitSW 監視
5	mgetty.conf	mgetty[2](モデム着信待受daemon)
6	mobile_watch.conf	MobileDevice(3G/LTE) アンテナレベル監視
7	shutdown-led.conf	shutdown時のLED設定
8	ttyO0.conf	getty (console が ttyO0 の場合)
9	ttyO1.conf	getty (console が ttyO1 の場合)

カスタムした Job の例

例として、シリアルコンソールの検出・起動を実現している console-detect.conf と ttyO0.conf(ttyO1.conf もほぼ同様) を見てみます。

console-detect.conf

```
description "Detect console device"

start on mounted MOUNTPOINT=/run

env consoledev
env CONSOLEDEV

emits consoledev

pre-start script
    for x in $(cat /proc/cmdline); do
        case $x in
        console=*)
```

```
            consoledev="${x#console=}"
            consoledev=${consoledev%,*}
            ;;

        esac
    done

    initctl emit --no-wait consoledev CONSOLEDEV=$consoledev
    stop
end script
```

ttyO0.conf

```
# ttyO0 - getty
#
# This service maintains a getty on tty1 from the point the system is
# started until it is shut down again.

start on stopped rc RUNLEVEL=[2345] and (
            consoledev CONSOLEDEV=ttyO0 and (
              not-container or
              container CONTAINER=lxc or
              container CONTAINER=lxc-libvirt))

stop on runlevel [!2345]

respawn
exec /sbin/getty -8 115200 ttyO0
```

処理の流れとしては、下記のようになっています。

- console-detect.conf
  - “/run” が mount されたイベントを契機に起動 (start on mounted MOUNTPOINT=/run)
  - 変数 “consoledev”, “CONSOLEDEV” を定義 (env ...)
  - イベント “consoledev” を定義 (emits ...)
  - pre-start のスクリプト実行 (/bin/dash による)
  - /proc/cmdline の内容をパースし、“console=ttyXXXX” から console を検出
  - initctl emit により“consoledev” イベントを発行 (パラメータ “CONSOLEDEV=ttyXXXX”)
- ttyO0.conf (ttyO1.conf)
  - 下記条件に当てはまる場合、処理が行われる (start on)
    - ランレベル [2345] の起動処理(rc)が終了
    - パラメータ “CONSOLEDEV=ttyO0” の consoledev イベントが発行されている
    - コンテナ内ではない(not-container)もしくは “CONTAINER=lxc”もしくは “CONTAINER=lxc-libvirt” である[3)]
  - ランレベル [!2345] (停止) の場合、停止する (stop on)
  - 処理が停止してしまった場合、再起動させる (respawn)
  - getty コマンドを実行する (exec /sbin/getty ...)

Job を定義する

簡単な例として、起動したら定期的にログを出力するだけのスクリプトを自動起動させる Job を定義してみます。
下記のような簡単なスクリプトを用意してみました。

sample.sh

```
#! /bin/bash

logger -t sample "Booted."

while true; do
  logger -t sample "alive."
  sleep 10
done
```

これを自動起動するための Job ファイルを作成します。/etc/init/sample-job.conf とします。

sample-job.conf

```
# sample job
#

description "sample user job"

start on runlevel [2345]
stop on runlevel [!2345]

respawn
exec /bin/bash /root/sample.sh
```

ためしに service コマンドで起動してみます。

```
root@plum:~# service sample-job start
sample-job start/running, process 1417
root@plum:~#
```

起動したようです。ps コマンドで確認してみます。

```
root@plum:~# ps ax |tail
 1213 pts/1    Ss     0:00 -bash
 1250 pts/1    S      0:00 sudo su -
```

```
 1257 pts/1    S      0:00 su -
 1266 pts/1    S      0:01 -su
 1338 ?        S      0:00 [kworker/0:2]
 1353 ?        S      0:00 [kworker/0:0]
 1417 ?        Ss     0:00 /bin/bash /root/sample.sh  <---
 1426 ?        S      0:00 sleep 10
 1427 pts/1    R+     0:00 ps ax
 1428 pts/1    R+     0:00 -su
root@plum:~#
```

起動していることが確認できました。
“respawn” オプションをつけておいたので、不意に死んだときに再起動されるか確認してみます。

```
root@plum:~# kill 1417
root@plum:~# ps ax |tail
 1210 ?        S      0:00 sshd: user1@pts/1
 1213 pts/1    Ss     0:00 -bash
 1250 pts/1    S      0:00 sudo su -
 1257 pts/1    S      0:00 su -
 1266 pts/1    S      0:01 -su
 1353 ?        S      0:00 [kworker/0:0]
 1445 ?        Ss     0:00 /bin/bash /root/sample.sh
 1448 ?        S      0:00 sleep 10
 1449 pts/1    R+     0:00 ps ax
 1450 pts/1    S+     0:00 tail
root@plum:~#
```

さきほどとは違うProcess ID (1417 ⇒ 1445) になって、起動していることがわかります。
明示的に停止させる場合は、service コマンドで制御します。

```
root@plum:~# service sample-job stop
sample-job stop/waiting
root@plum:~# ps ax |tail
 1191 ?        Ss     0:00 sshd: user1 [priv]
 1210 ?        S      0:00 sshd: user1@pts/1
 1213 pts/1    Ss     0:00 -bash
 1250 pts/1    S      0:00 sudo su -
 1257 pts/1    S      0:00 su -
 1266 pts/1    S      0:01 -su
 1353 ?        S      0:00 [kworker/0:0]
 1455 ?        S      0:00 [kworker/0:2]
 1476 pts/1    R+     0:00 ps ax
 1477 pts/1    S+     0:00 tail
root@plum:~#
```

詳細情報

より細かい制御を行いたい場合、公式ページの Cookbook を参照してください。

1)
Webベースターミナル(butterfly)の利用
2)
http://mgetty.greenie.net/
3)
container-detect.conf で発行されるイベントです (not-container/container)