lcfg-nut overhaul for SL7

This is the final report for Project 375 - lcfg-nut overhaul for SL7.

Contents

1. Introduction
2. Outcome of the project
3. General comments
4. Future work
5. Effort
Appendix A - UPS power control for self-managed servers
   A.1 Motivation
   A.2 Current Forum server room UPS setup
   A.3 Design intent of NUT
   A.4 Requirements for UPS power control for self-managed servers
      A.4.1 From our point of view
      A.4.2 From the users' point of view
   A.5 A suggested NUT configuration to incorporate self-managed servers
      A.5.1 Self-managed servers
      A.5.2 Virtual UPS master
   A.6 Summary

1. Introduction

The initial aims of this project were:

1. to port the lcfg-nut component to SL7; and
2. in the course of that work, to overhaul the component as deemed either necessary or desirable.

Subsequent to the initial aims, further work was identified in relation to UPS power control for self-managed servers. This is fully discussed in Appendix A.

2. Outcome of the project

1. The lcfg-nut component and associated headers have been ported to SL7, and are in active use.
2. A viable scheme for UPS power control for self-managed servers was developed and prototyped - see Appendix A. However, actual use of the suggested scheme is now completely mothballed: what we can/should/will do in future will depend entirely on the actual future details of UPS implementation in the Informatics Forum - and these are currently in flux.

3. General comments

1. The port of lcfg-nut to SL7 went smoothly enough after some time spent in initial orientation: only minor changes to the component proved to be necessary.
2. The SL7 port of the component and headers uses a locally built version of the most recent release (2.7.4 at the time of the work) of the NUT software. This is the same approach taken as for previous versions of the component, but falls short of one of the initial aspirations of this project - namely, to make use of stock RPMs.
3. RPMs for NUT (currently at 2.7.2) are indeed available in EPEL. If we do want to use those in favour of locally-built versions, we will need to make further changes to the component and headers (e.g. fully embrace systemd for NUT management; configure standard directory locations; etc.) Arranging all that has now been postponed to a future effort.
4. The port of lcfg-nut to SL7 was completed a long time (about a year?) ago; what has held this project up since has been the uncertainty of future UPS provision for the Forum server room, and the implications which that has on the work described in Appendix A. We now propose to draw a line under all that, and to revisit that matter when the overall situation has become clearer.
4. Future work

1. Submit a ‘wishlist’ project for lcfg-nut overhaul: any such presumably to be done in time for (or as part of) the SL8 OS upgrade.

   *Comment:*
   
The lcfg-nut component is visibly old; and it would certainly would not take its current form were it to be written from scratch today. However, it works; and its size and overall complexity means that any redesign and rewrite will not be a trivial matter. We need to be clear about why we are suggesting any rewrite; and we need to be clear about the detailed objectives of any such rewrite. If some of the impetus for a rewrite is connected with, say, a desire to convert the component from bash to Perl, or a desire to fully embrace the use of systemd, then it seems to me that those same considerations would apply to many other LCFG components here - in which case the full 'project' might presumably be to rewrite all such components.

   Meanwhile, I have submitted my own suggestions regarding a rewrite of lcfg-nut via other channels.

2. Implement and advertise a scheme for UPS power control for self-managed servers. (See Appendix A.)

5. Effort

The total effort for this work was approximately 15 days.

---

Appendix A - UPS power control for self-managed servers

A.1 Motivation

Currently, both the main Forum server room and the self-managed server room are powered via the same UPS system (which actually consists of a pair of UPSes.) This means that, should a power failure occur, the total run time available to 'School' servers is less than it would be were self-managed servers not involved.

This would seem fine if, first, the UPS system involved offered a reasonable long run time; and, second, if the overall power draw of the self-managed servers was modest in relation to that of School servers. However, the run time of the UPS is now quite short (possibly as short as 15 minutes); and the overall power draw of the self-managed server room is (at the time of writing) comparable to that of the main server room (see a snapshot graph of the Forum aggregate power usage at 2017-05-04.) So, given that we would like - so far as possible - to protect crucial School servers at times of power failure, the question arises of how we should handle self-managed servers at such times.

The current approach is that members of the Computing staff will manually power down self-managed servers at times of power failure. That is not a very satisfactory situation, so a decision was taken to extend the current project to investigate the possibility of power-down automation via NUT.

A.2 Current Forum server room UPS setup

The server room UPS consists of a pair of UPSes. Machines in the main server room monitor both UPSes by polling their controlling servers over the network. (In turn, those controlling servers poll the UPSes directly via SNMP.) The shutdown condition (as managed by the `upsmon` process, and configured by appropriate settings of NUT’s `MINSUPPLIES` and `powervalues` parameters) is that both UPSes have reached the ‘ON BATTERY; LOW BATTERY’ state.

(Note that currently (and since about September 2016?) one of the pair of UPSes is in a faulted state, so machines in the main server room currently only monitor the single healthy UPS. There are current plans either to completely replace the server room UPS system, and/or to repair the failed unit - but actual progress on those plans is unknown.)

A.3 Design intent of NUT

The design intent of NUT is that users should 'trust' the system, and should accept that a shutdown will be instructed by the watching `upsmon` process if (and only if) the UPS being monitored reaches an ‘ON BATTERY; LOW BATTERY’ state.
It is possible to take action on other NUT state transitions using NUT’s `upssched` mechanisms, but doing so has a rather ‘manual’ feel about it, and seems to be discouraged.

### A.4 Requirements for UPS power control for self-managed servers

#### A.4.1 From our point of view

We would like self-managed servers to be shut down long before both of the server room UPSes reaches their ‘ON BATTERY; LOW BATTERY’ state. Our ad-hoc proposal is that in fact such a shutdown should be initiated when both UPSes have been in the ‘ON BATTERY’ state for more than 30 seconds.

#### A.4.2 From the users’ point of view

Any NUT configuration we suggest should be:

1. As simple, clear, standard, and easy to implement as possible. (We will be asking users to implement such a configuration on their own machines.)
2. As robust as possible. (Specifically, we must avoid any possibility of false alarms: should any user suffer any problems as a result of our suggested configurations then they will be understandably reluctant to use our suggested configuration in future)
3. Fixed once-and-for-all, and not subject to routine change. (We cannot expect users to keep up with any frequent changes we might want to make.)

Should we achieve this, then we get the benefit of more power for the School’s servers at times of power supply problems, and owners of self-managed machines get the benefit of predictable and controlled shutdowns of their machines at such times.

### A.5 A suggested NUT configuration to incorporate self-managed servers

Taking account of both A.3 and A.4, we suggest the following model:

1. A master school server which is implemented as a VM, and which:
   1. Monitors both of the actual server room UPSes.
   2. Uses the `dummy-ups` driver supplied by NUT to implement a single virtual UPS.
   3. Uses `upssched` to trigger a shutdown of that virtual UPS in the event that both server room UPSes have been in the ‘ON BATTERY’ state for more than 30 seconds.
2. All self-managed servers to monitor the single virtual UPS presented by the above.

Configuration to implement this scheme is more-or-less as follows:

#### A.5.1 Self-managed servers

```
================================== upsmon.conf ==================================
MONITOR smsr-ups@smsr-ups-master.inf.ed.ac.uk 1 <username> <password> slave
SHUTDOWNCMD "/sbin/shutdown -h +0 "UPS low battery\""
NOTIFYFLAG ONLINE SYSLOG
NOTIFYFLAG ONBATT SYSLOG
NOTIFYFLAG LOWBATT SYSLOG+Wall
NOTIFYFLAG FSD SYSLOG+Wall
...
```

#### A.5.2 Virtual UPS master

```
================================== upsmon.conf ==================================
```
MONITOR snmp50@<ups50master>.inf.ed.ac.uk 0 <username> <password> slave
MONITOR snmp51@<ups51master> 0 <username> <password> slave
SHUTDOWNCMD ""/sbin/dummyshutdown-cmd.sh"
NOTIFYFLAG ONLINE SYSLOG
NOTIFYFLAG ONBATT SYSLOG
NOTIFYFLAG LOWBATT SYSLOG+WALL
NOTIFYFLAG FSD SYSLOG+WALL

...[snip]...

============================ ups.conf =============================

[smr-ups]
  driver = dummy-ups
  port = /var/lcfg/conf/ups/smr-ups.dev
  desc = "Self-manager server room virtual UPS"

============================ upssched.conf =============================

CMDSCRIPT /var/lcfg/conf/ups/upssched-cmd.sh
PIPEFN /var/lcfg/conf/ups/upssched.pipe
LOCKFN /var/lcfg/conf/ups/upssched.lock

AT ONBATT * START-TIMER onbatt 30
AT ONLINE * CANCEL-TIMER onbatt

============================ upssched-cmd.sh =============================

#!/bin/sh
#
# This script is called by upssched via the CMDSCRIPT directive.
#
# The first argument passed is the name of the timer from the corresponding 'AT' line.

UPSC=/usr/ups/bin/upsc
set -o noglob

case $1 in
  *)
    logger === upssched-cmd $1 ===
    logger ** `id`
    snmp50status=`$UPSC snmp50@<ups50master>.inf.ed.ac.uk | grep ups.status`
    snmp51status=`$UPSC snmp51@<ups51master>.inf.ed.ac.uk | grep ups.status`
    logger ** snmp50@<ups50master>.inf.ed.ac.uk: $snmp50status`
    logger ** snmp51@<ups51master>.inf.ed.ac.uk: $snmp51status`
    echo $snmp50status | grep -q 'OB' && echo $snmp51status | grep -q 'OB' && logger ** BOTH UPSes ON BATTERY && upsmon -c fsd
    ;;
    esac

============================ dummyshutdown-cmd.sh =============================

4 of 5 30/05/17 11:05
#!/bin/bash

echo === $0 - killing upsd ====
killall upsd

echo === $0 - stopping upsdrvctl ====
/usr/ups/sbin/upsdrvctl stop

echo === $0 - sleeping ... ====
sleep 10

echo === $0 - starting upsdrvctl ====
/usr/ups/sbin/upsdrvctl start

echo === $0 - starting upsd ====
/usr/ups/sbin/upsd

Note that the master server never actually shuts itself down in response to power failure: in the case of a real power outage, we expect that it will be shutdown under the command of the KVM server on which it resides. The purpose of the dummyshutdown-cmd.sh command is to restore the virtual UPS to functionality after a shutdown of that UPS has been instructed. If this is not done, the system will not be robust in the case where, for example, power fails for 31 seconds, but is then restored. Were action not taken to restart the virtual UPS, such a sequence of events would mean that the virtual UPS was not then subsequently available for monitoring by self-managed servers.

### A.6 Summary

The scheme presented in section A.5 has been tested so far as is possible (*), and seems to work as expected. However it's obviously slightly ad-hoc, is not easy to test against a real case of power failure; and feels somewhat fragile.

(* Testing involved, amongst other things, reimplementing the snmp50@<ups50master> and snmp51@<ups51master> UPSes as local virtual UPSes, and then commanding various 'ON BATTERY' etc. conditions in those virtual UPSes via the upsrw command.)

I suggest that, before we finally implement anything in this regard - and, specifically before we ask the owners of self-managed servers to implement anything -, we wait until the situation regarding overall server room provision has been properly clarified. Currently, we really don’t know what we’re dealing with. The best final position to end up with would be a satisfactorily-specified single server room UPS which both School and self-managed servers could monitor in an entirely standard way.

-- IanDurkacz - 27 Apr 2017