DKUUG-NYT

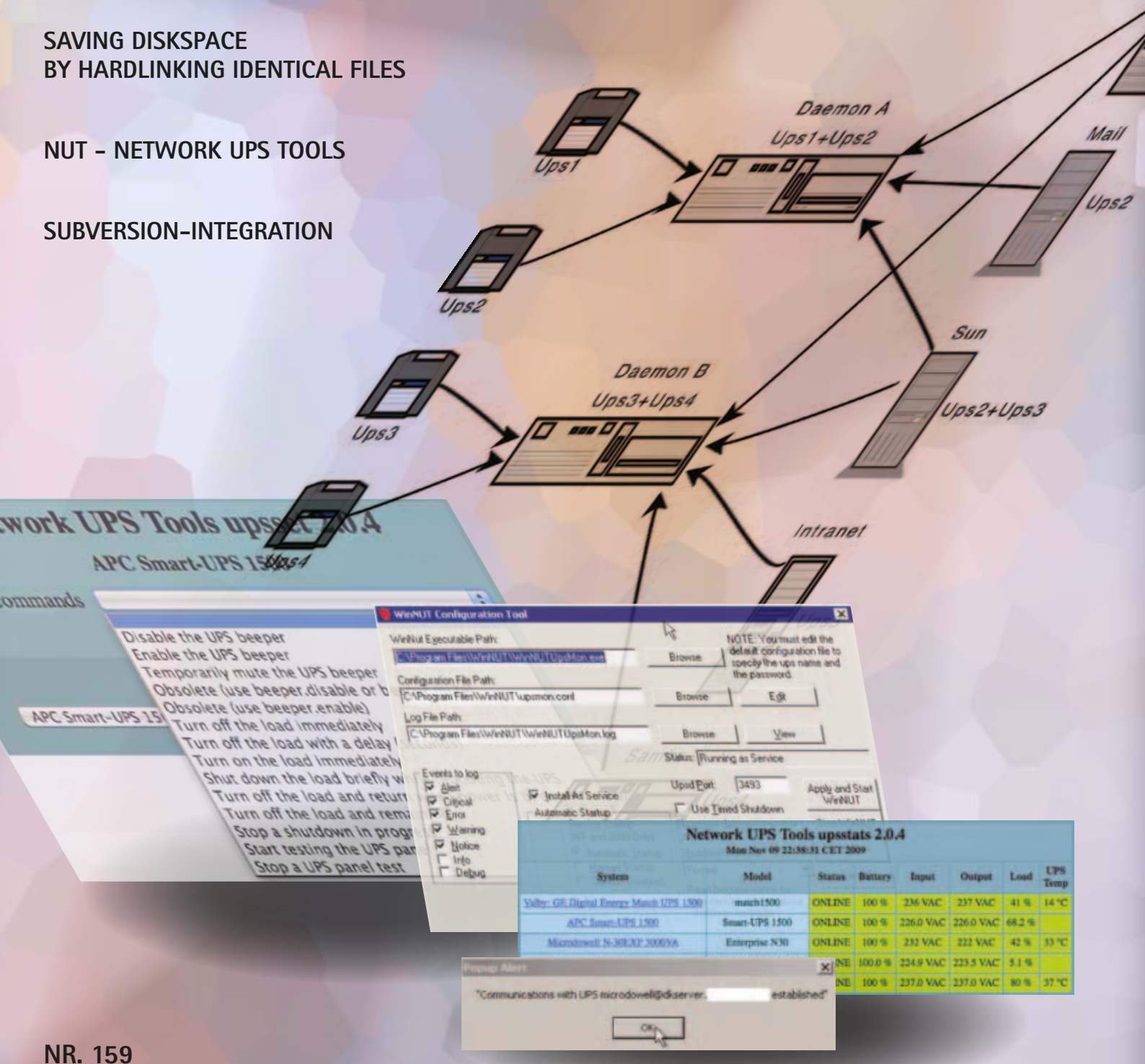
HVAD VIL DKUUG I NÆSTE ÅRTI

SÅDAN TAGER JEG BACKUP

SAVING DISKSPACE
BY HARDLINKING IDENTICAL FILES

NUT - NETWORK UPS TOOLS

SUBVERSION-INTEGRATION



NR. 159
NOVEMBER 2009

HVAD VIL DKUUG I NÆSTE ÅRTI?

Vi står foran noget, som af de fleste opfattes som skift fra et årti til et andet, selv om de kloge hævder, at årtiet varer til 31. december 2010 :-).

Ved årsskiftet 1999-2000 opstod der næsten panik, ville fly falde ned og hospitaler gå i stå, fordi tid og dato pludselig så helt anderledes ud? Ville det betyde tab af menneskeliv?

En større skare af IT-konsulenter fik et levebrød af at se på datoernes implementering i forskellige systemer. Det er jo sådan set meget godt, men var de penge og uddannelseskræfter den bedste anvendelse af ressourcerne?

Den situation med datoerne omkring årtusindskiftet viste, hvor meget uvidenhed, der er omkring computersystemer, og hvor vigtigt det er, at man kender kildekoden til sine systemer, så man kan inspicere og se, om fx. datoer er repræsenteret på en måde, der får 2000-01-01 til at være mindre end 1999-12-31, hvad der jo er tilfældet hvis man skriver 99-12-31 og 00-01-01 (Det gjorde bl.a. visse Cobol implementationer).

Ud over nogle seriøse fejl med fx. renteberegning med en dårlig dato-implementering og indkaldelse af 100-årige til børnehave, så var der ikke alvorlige uheld ved årtusindskiftet. At det alligevel fik så stor en omtale, hænger måske sammen med at talmagi kombineret med dommedagsprofetier stadig kan fylde spalter i aviserne fordi det både er morsomt og uhyggeligt, når man er lidt usikker på, hvad de her computere egentlig er for nogen størrelser.

Når vi nu skifter til næste årti, er der ikke samme panik-stemning indenfor det område. Forventningerne til det nye årti går først og fremmest på, at de politiske kræfter vil arbejde mere sobert på styring af finanssektorerne og på udnyttelsen af fossile brændstoffer og andre geopolitiske spørgsmål. Computer-magi er trådt i baggrunden.

De slagsmål, som udkæmpes om klimabevarende foranstaltninger og om styringen af finanssystemerne overskygger langt slagsmålet om anbefaling af standardisering indenfor computernetværk, digitale dokumenter og operativsystem API'er. Det er naturligt nok - computersystemer skal i de fleste tilfælde blot være hjælpesystemer.

Imidlertid vil der, i et levende samfund med markedsmekanismer, hele tiden være en kamp om markedsandele, som udkæmpes ved at bruge tekniske finesser. Vi kender alle en netadministrator, som siger "det er meget lettere med produkt Xyzzy" og talemåden at "ingen er blevet fyret for at købe XX".

Hvis den tendens ikke holdes i skak, vil vi få et marked, som bliver mere og mere uigennemsigtigt, og som styres af private interesser i stedet for fælles interesser.

Hvis ikke der er en interesseorganisation, som kan gøre opmærksom på farerne ved ineffektiv og dyr monopolisering, så vil den teknisk ukyndige leder tage beslutninger, som medfører spild af ressourcer. Det har vi set mange eksempler på i det offentlige og mindre publiceret i det private erhvervsliv.

Der er mange flere spørgsmål, som kræver bredere orientering. Det er en kendt ting, at Open Source miljøet har en vis skepsis overfor de store leverandører, men vi, der taler varmt for brug af Open Source bør kunne forstå de brugere, som har svært ved at skifte fra en form for klikbarhed til en anden.

Vi bør kunne forklare, at det ikke altid er rimeligt at "nemmere brugerbetjening", "centralt overblik over licenser" m.v. bruges som mantra for indførelse af nye proprietære systemer. Især "nemmere brugerbetjening" bruges ofte som årsag til at fx. mailklienter automatisk skal kunne starte andre programmer, og det var årsagen til at "I Love You" virussen og dens efterkommere spredtes over hele kloden.

Bladet indeholder denne gang artikler, som viser, hvordan koncern backup kan bygges på avanceret vis ud fra Open Source værktøjer. Desuden bringer vi en beskrivelse af, hvordan UPS kan håndteres af Linux servere.

Kenneth Geisshirt har skrevet en artikel om subversion, der kan bruges til både revisionsstyring og dokumenthåndtering. Det er noget, som vi gerne vil kunne give medlemmerne et real-life eksempel på, og når vi får vores virtuelle servere at køre, kan vi i DKUUG bruge en SVN server til gruppe-samarbejde.

Det kan ofte være et problem for begyndere, at Linux dokumentationen er på engelsk og er spredt over flere hundrede websites. Der er virkelig noget at gøre for en forening, som vil samle viden og systematisere den. Selv om der via foreningen SSLUG er gjort et kæmpearbejde på dansk, kommer der stadig nye ting på markedet, som kræver mere omtale. Dette bør være en opgave for Open Source community'et og dermed også for DKUUG.

Community Day d. 24. oktober ser ud til at have løbet rundt. Der var stor interesse for eventen. Du kan finde billeder og slides på http://www.opensourcedays.org/CommunityDay2009/.

Foruden foredragene var der blandt andet to BOF'er, en om brugergrupperne i Danmark og deres samarbejde og en om organiseringen af OSD.

Det skriver vi mere om i næste blad. Foreløbig kan vi konstatere, at der er et behov for dage med foredrag for fagfolk om Open Source teknik. OSD har gjort et fremragende stykke arbejde og er begyndt at planlægge for 2010.

Jeg overtog formandsskabet for DKUUG i 2006. I 2008 (sidste år, hvis nogen skulle have glemt det :-)) gav jeg med glæde faklen videre til Svenne Krap, som for nyligt holdt et foredrag om OpenPGP som sikkerhedsforanstaltning på nettet.

Mine visioner for DKUUG var at en ung generation skulle deltage i genopbygningen af en levende organisation med sjov og underholdende oplysning og med fagligt velfunderede medlemmer, som kan tage ordet i den offentlige debat.

Vi har stadig brug for den forening, så støt op omkring den unge generation.

Donald Axel

INDHOLD

DKUUG-Nyt er medlemsbladet for DKUUG, foreningen for Åbne Systemer og Internet

Udgiver:
DKUUG
Fruebjergvej 3,
2100 København Ø
Tlf: 39 17 99 44
Fax: 39 20 89 48
email: dkuugnyt@dkuug.dk

Redaktion:
Jon Bendtsen (ansvarshavende)
Donald Axel
Kristen Nielsen
Dennis D. Jørgensen

Tryk:
Typographic ApS

Design og Layout:
Seifert Design & Grafik

Annoncer:
Kontakt DKUUGs sekretariat
sek@dkuug.dk

Oplag:
500 eksemplarer

Medlem af Dansk Fagpresse

DKUUG-Nyt
ISSN 1395-1440

SÅDAN TAGER JEG BACKUP

AF JON BENDTSEN - jonbendtsen@jbit.dk

I denne artikel vil jeg fortælle hvordan jeg tager backup for en af mine kunder. Dele af teksten er på engelsk fordi det er koncernsproget, og jeg har lavet teksten som en del af en intern dokumentation af systemernes opsætning.

Backupsystemet er bygget op af 2 backup servere med en masse diske. Disse maskiner kører debian Linux med sshd som eneste service. De henter deres data fra nogle Linux servere og nogle windows servere.

Data overføres med rsync hen over ssh for Linux serverne, og bare direkte med rsync for windows maskinerne. På windows serverne bruger jeg cwrsync server fra http://www.itefix.no/i2/cwrsync

Jeg har selvfølgelig prøvet at lave en fuld restore, det tog 3 døgn, så derfor har jeg en hotspare maskine af den vigtigste server.

Da rsync bare lægger filerne enkeltvis så er det meget nemt at lave restore på enkelte filer, og ofte kan jeg nemt sammenligne de forskellige udgaver af backupen fra forskellige datoer og give folk den fil som passer dem bedst.

To avoid copying unchanged data, the backupscripts starts by doing a hardlink copy of the backup from yesterday. But first we make a new directory for today using:

```
daynr=$(date +%j_%A_%d)
mkdir /rsyncbackup/daily/$daynr
```

The hardlink copy is done like this:

```
yestday=$(ls -t /rbackup/daily/ | head -2 | tail -1)
cp -al /rbackup/daily/$yestday/* /rbackup/daily/$daynr/
```

We are now ready to do a rsync into /rbackup/daily/$daynr. This rsync will run with --delete and will then automatically delete files that no longer exists on the backupclient. Further more rsync will actually delete existing but changed files and then write a new file with todays changes using the same name. This means that identical and unchanged files will be hardlinked against those from yesterday, thus saving alot of diskspace, because only the changed files take up new space.

Once the rsync is done for all backupclients, then we test if it is time to do a weekly, monthly and/or yearly backup. If it is time to do then we do another hardlink copy to the right place.

```
DST="/rbackup/weekly/$weeknr"
today=$(ls -t /rbackup/daily/ | head -1)
cp -al /rbackup/daily/$today/* $DST/
```

To avoid filling up the disks we then delete old backups. We keep 5 weekly and 12 months. The yearly are never deleted but kept forever. Or at least so far forever. The oldest backup are deleted with a simple:

```
rm -rf $(ls -t /rbackup/weekly/| tail -n +6)
```

My scripts give me some more statistics, like number of deleted files, transferred files and total files. I also get to know how much disk is used before and after each step. With this I can keep an eye on how fast the backup grows. The scripts use lock files to ensure only one of them is running at any given time.

I use one script to do the setup and copy the data from yesterday. Then I use a separate script for each server to transfer the data from the servers. Finally I have a weekly, a monthly and a yearly backup script.

DATA RETENTION STRATEGY

The backup data is organised with a top directory like this:

```
daily/
weekly/
monthly/
yearly/
```

Diving into the daily/ directory we see:

```
069_Tuesday_10/
070_Wednesday_11/
074_Sunday_15/
075_Monday_16/
076_Tuesday_17/
latest/
```

latest is a symbolic link that points to the latest directory, in this example it would be to 076_Tuesday_17/. Having the latest directory also tells me that the backup from yesterday was completed without errors. The first 3 numbers are the day of the year. The last number is the day of the month.

If we dive further into any of these days, then each backupclient has a directory with the name of the backup client:

```
mail/
samba/
intranet/
```

And inside those directories are the data from the backupclients. Usually from / with /proc /dev/ ... excluded. But from windows servers it is just a copy of the wanted data.

If we go back to the top directory, and look into weekly we see:

```
05_Thursday_29/
06_Thursday_05/
07_Thursday_12/
08_Thursday_19/
09_Thursday_26/
```

The first 2 numbers are the weeknumber, the last 2 numbers are the day of the month. Notice how all are taken on Thursday. The other backup server takes weekly backup on a different weekday, Saturday. The reason is to have more backups to choose from if I need to restore. Below those directories are the same names of the backupclients as there was in daily/

Looking into monthly we see:

```
April_2008_16/
May_2008_16/
June_2008_16/
July_2008_16/
August_2008_16/
September_2008_16/
October_2008_16/
November_2008_16/
December_2008_16/
January_2009_16/
Februar_2009_16
March_2009_16/
```

The year and the day of month. As with the weekly, the other backup server will have a different day of the month to take monthly backup. And those directories contain the same as the weekly and daily directories.

Diving into the yearly we see:

```
2006_July_01/
2007_July_01/
2008_July_01/
```

Again the other backup server has a different day to take a yearly backup. The other server is January_01/

One full backup seems to currently take about 2.5TB, this means that all the rest of the 7.6TB data is the different versions. If i didnt "compress" the disk usage by hardlinking identical files, the data would fill something like 2.5TB * (12 months + 5 weeks + 5 days + 3 years) or 25 times. Now i get away with just 3 times. Not a bad "compression".

Another advantage of only copying the changed data is that the backup does not strain the system so much, and it does not take so long time.

My backup system has been upgraded since i started working for them, and even moved to a new computer, but it basicly still uses the same scripts, setup and layout.

Diskbased backup may not be as failsafe as tape, but it does seem more cost effictive, and it is probably a lot faster to restore.

SAVING DISKSPACE BY HARDLINKING IDENTICAL FILES

Because my users move data all over the place, and because I sometimes migrate the backup data to a new server, then I occasionally run scripts that finds and hardlinks all identical files.

Obviously identical files will all be of the same size, and therefore I "only" need to find all files of a given size.

But when you have 7.6TB data on 18,136,792 inodes, then you get a lot of different sizes. To keep track of all these filesizes I ended up creating a file for each filesize. Naturally the name of this file is the filesize.

This allowed me to create 2 scripts. 1 script finds all files, their sizes and puts this into files based on the filesize. Then a 2. script handles each file one after another.

The first find script was a simple bash script where I piped find output into.

```
find -type f -printf "%i %s %p\n" |\
  ./split_files_by_size.sh files_by_size/
```

```
dksrv003:/rbackup# cat split_files_by_size.sh
#!/bin/bash
if [ "$1" == "" ]; then
 echo "This script needs a directory argument"
 exit 1
fi

if [ -d "$1" ]; then
 # do my magic here
 read line
 until [ "$line" == "" ]; do
  # all lines are built like this "inode size path"
  size=$(echo $line | cut -d" " -f2 | sed -e "s/ //g")
  echo "$line" >> "$1/$size"
  read line
 done
else
 echo "This script needs a directory as an argument"
 exit 2
fi
```

But this method is rather slow, possibly because read line is slow.

I made a faster version in python:

```
dksrv003:/rsyncbackup# cat find_inode_size_name.py
#!/usr/bin/python
#
#  find_inode_size_name.py
#
#
#  Created by Jon Bendtsen on 17/10/08.
#

import os
import stat
import sys
import string

if 3 != len(sys.argv):
 print "usage: ./find_inode_size_name.py startdir datadir"
 exit(1)

startdirectory = sys.argv[-2]
datadirectory = os.path.abspath(sys.argv[-1])

top=startdirectory
or root, dirs, files in os.walk(top, topdown=False):
 for name in files:
  fullname=os.path.join(root, name)
  if True == os.path.isfile(fullname):
   size=os.stat(fullname)[stat.ST_SIZE]
   inode=os.stat(fullname)[stat.ST_INO]
   f=open(datadirectory + "/" + str(size),"a+b")
   f.write(str(inode) + " " + str(size) + " " + fullname + "\n")
   f.close()
```

The output of both of the find scripts above is a lot of files, one for each data filesize. Each file contains 1 line for each data file with the same size. So for size 99 bytes, I have 38344 lines because there are 38344 files with the size 99 bytes.

Examples files are:

1 2 7 17 99 10987 26291 40950859344 ...

Each line is build up like this:

inodenumber size fullpath

```
287425755 99 ./monthly/October_2008_16/dk2/usr/lib/aspell/split.kbd
289066745 99 ./monthly/October_2008_16/dk2/usr/lib/python2.4/...
289198726 99 ./monthly/October_2008_16/dk2/usr/share/debhelper/...
289204529 99 ./monthly/October_2008_16/dk2/usr/share/menu/bc
289204530 99 ./monthly/October_2008_16/dk2/usr/share/menu/dc
298091155 99 ./monthly/October_2008_16/dk2/var/lib/vservers/...
```

The inodenumber is used to keep track of identical files which are already hardlinked together.

The 2. script operates on each of these files. This script is a bash script. The 2. script parses the files the 1. script makes. It extracts the inode number and the fullpath. If the inodes are identical, the files are already hardlinked together and it quickly moves on to the next line. If inodes are different it checks the fullpath for existance and then starts comparing if they are identical. It exits at the first different bit. Through a double for loop each line and thus each file is compared to every other file.

Once it finds 2 identical files it hardlinks them together + any other lines which has the same inode number. Thus if there are 2 chunks of lines which has the same inode number these are not compared to each other inside the chunks, but once just one of the lines in the 2 chunks are identical both chunks are turned into one new chunk all hardlinked together.

```
[same_size_cmp_freshfile_hardlink.sh  text/plain (3.2KB)]
#!/bin/bash

OLDIFS=$IFS
IFS="
"

realfile="$1"
file=$(tempfile --suffix same_size_cmp_hardlink.sh-file)
cat $1 | sort -k1,1 -u >> $file

compareandhardlink() {
# we have one more check if it is the same inode, because we
# REALLY do not want to check if they are
if [ "$iname" != "$xname" ]; then
 nodei=$(ls -i "$iname" | cut -d" " -f1)
 nodex=$(ls -i "$xname" | cut -d" " -f1)
 if [ "$nodei" -ne "$nodex" ]; then
  cmp -s "$iname" "$xname"
  if [ 0 -eq $? ]; then
# okay, so they are identical, but the xname file might itself have
# other hardlinks. Lets find them all and all hardlink them to iname
    echo "  Files:"
    ls -lai "$iname"
    ensfiler=$(tempfile -suffix same_size_cmp_hardlink.sh-ensfiler)
    grep "^$nodex " $realfile > $ensfiler
    grep "^$xinode" $realfile >> $ensfiler
    for ens in $(sort -u -k3 $ensfiler); do
     ensname=$(echo $ens | cut -d" " -f3-)
     if [ -f "$ensname" ]; then
       ensinode=$(ls -i "$ensname" | cut -d" " -f1)
       if [ "$nodex" -eq "$ensinode" ]; then
         ls -lai "$ensname"
         ln -f "$iname" "$ensname"
         if [ 0 -ne $? ]; then
                echo "9 - linking failed, something is wrong here, check file $realfile manually"
                mkdir -p files_with_error
                mv "$realfile" files_with_error exit 9
              fi
```

```
             elif [ "$nodei" -eq "$ensinode" ]; then
              ls -lai "$ensname"
             else
              cmp -s "$iname" "$ensname"
              if [ 0 -eq $? ]; then
                   ls -lai "$ensname"
                   ln -f "$iname" "$ensname"
                if [ 0 -ne $? ]; then
                    echo "11 - linking failed, something is wrong here, check file $realfile manually"
                    mkdir -p files_with_error
                    mv "$realfile" files_with_error exit 11
                fi
               else
                   echo "$ensname" >> checkfilesmanually.txt
               fi
              fi
             fi
            done
            rm -rf $ensfiler
            echo "    are identical and now also hardlinked together"
            echo ; echo
          fi
         fi
        fi
      }

maxlines=$(wc -l < $file)
if [ "1" -eq "$maxlines" ]; then
          rm -rf "$file"
          rm -rf "$realfile"
          exit 3
fi
#echo "$realfile" >> /tmp/countfile2
for ((i=1;i<=$maxlines;i++)); do
        li=$(sed --silent -e "$i p" $file)
        for ((x=$i;x<=$maxlines;x++)); do
#             echo -e "\t$i\t$x" >> /tmp/countfile2
# if you run cmp with the same file or inode it will still report
# the same exitcode as if the files are just identical. Thus if
# we delete the "second" file we also delete the "first" file, so
# i check in advance to make sure cmp will be run with non identical
# file names.
               if [ "$i" -ne "$x" ]; then
                        lx=$(sed --silent -e "$x p" $file)
# there is no need to calculate liinode before I need it
                        iinode=$(echo $li | cut -d" " -f1)
                        xinode=$(echo $lx |  cut -d" " -f1)
# Just as above, it might be bad to run cmp with the same inode.
# besides it is probably a waste of resources to start an external
# program. So we compare the inode here, and only if they are
# different will I compare the 2 files.
                         if [ "$iinode" -ne "$xinode" ]; then
# there is no need to calculate iname or xname before I need it
                              iname=$(echo $li | cut -d" " -f3-)
                              xname=$(echo $lx | cut -d" " -f3-)
# we are now ready to compare the lines
                              if [ -f $iname ]; then
                                  if [ -f $xname ]; then
                                      compareandhardlink
                                  fi
                              fi
                        fi
               fi
       done
done

rm -rf "$file"
rm -rf "$realfile"
```

NUT – NETWORK UPS TOOLS

SÅ DINE SERVERE LUKKER PÆNT NED FØR UPS'EN LØBER TØR

AF JON BENDTSEN - jonbendtsen@jbit.dk

Network UPS Tools upsstats 2.0.4
Mon Nov 09 22:38:31 CET 2009

System	Model	Status	Battery	Input	Output	Load	UPS Temp
Valby: GE Digital Energy Match UPS 1500	match1500	ONLINE	100 %	236 VAC	237 VAC	41 %	14 °C
APC Smart-UPS 1500	Smart-UPS 1500	ONLINE	100 %	226.0 VAC	226.0 VAC	68.2 %	
Microdowell N-30EXP 3000VA	Enterprise N30	ONLINE	100 %	232 VAC	222 VAC	42 %	33 °C
Norways: HP R3000 XR UPS	HP R3000 XR 3000VA	ONLINE	100.0 %	224.9 VAC	223.5 VAC	5.1 %	
Switch: Microdowell BP-500	500VA	ONLINE	100 %	237.0 VAC	237.0 VAC	80 %	37 °C

Når man har servere i produktionsdrift så er det rart at vide om de har strøm nok. Da man ofte sjældent opholder sig i serverrummet, er det ikke nok at kunne høre den alarm som UPSen laver. Desuden så kan man have så mange servere at man alligevel ikke vil kunne nå at lukke dem manuelt ned. Derfor så findes der software som overvåger UPSen og lukker serveren pænt ned før batteriet er løbet tør. Ofte leveres denne software sammen med UPSen, men den kan kun snakke med den ene fabrikants UPS og måske endda kun med den ene model af UPSe.

Selvom man har mere end 1 server, så har man sjældent 1 UPS pr. server, men ofte flere servere pr. UPS. Ofte er overvågnings forbindelsen på en UPS en seriel eller USB forbindelse, og så er det svært at forbinde flere servere til den samme UPS. Det kunne løses med en netværks port, men det kommer "små" UPSe sjældent med.

Løsningen er en central netværks service som overvåger en eller flere UPSe som er fysisk forbundet til 1 server. Alle de andre servere forbinder sig så til denne netværks service, og får via denne status på UPSene at vide.

En udgave af en sådan software hedder Network UPS Tools og forkortes NUT. Denne artikel vil omhandle konfigurering af NUT, med udgangspunkt i den opsætning som jeg har lavet for en af mine kunder.

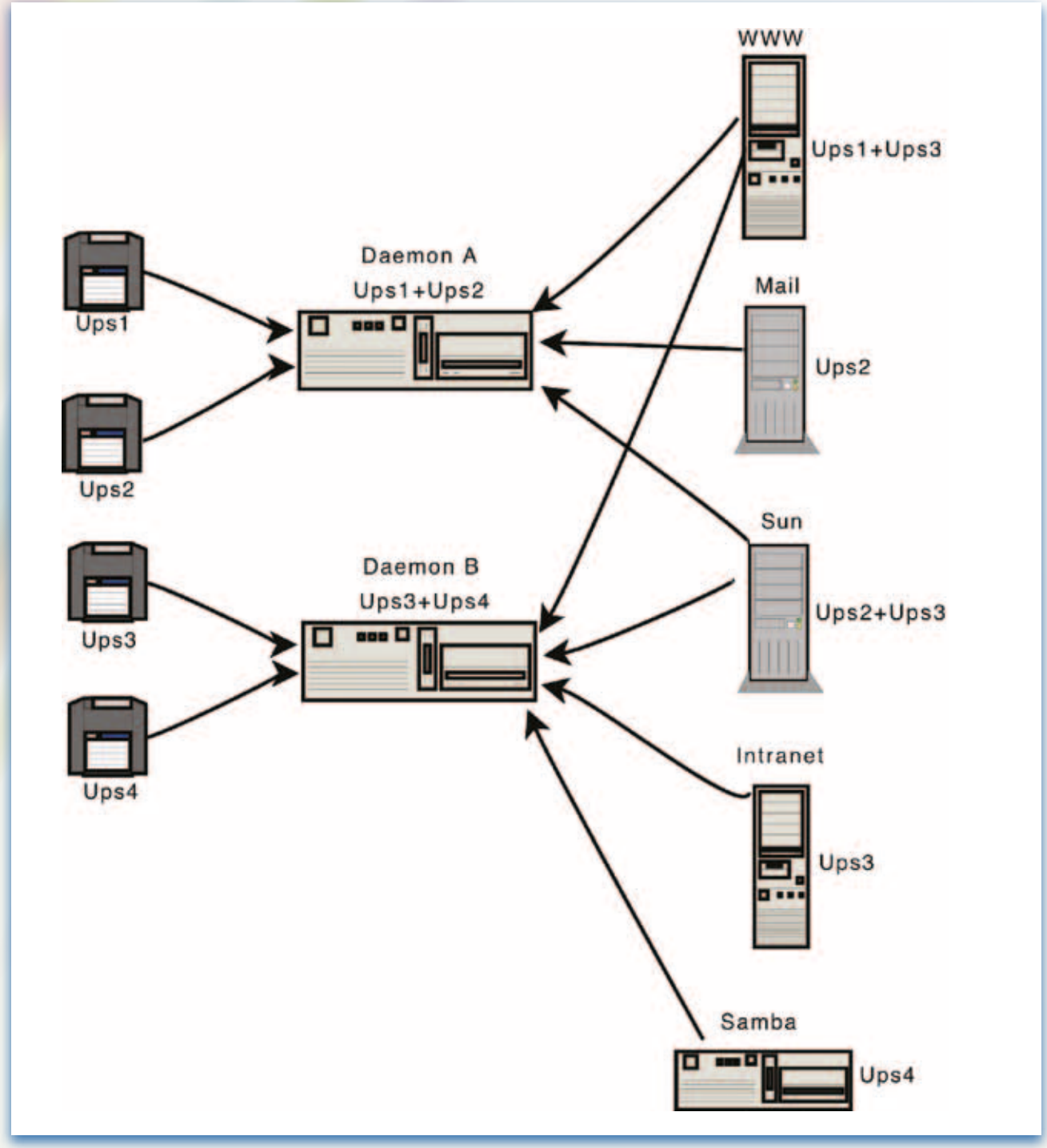
NUT er et open source projekt frigivet under GPL som kan findes på adressen: http://www.networkupstools.org/.

NUT består af 3 komponenter, UPS driver, UPS daemon og UPS monitor.

UPS driveren kører på den maskine som er fysisk forbundet med en UPS enten via USB eller også via serielt.

UPS daemon kører på den samme maskine, men som en separat process, og der kører kun 1 UPS daemon selvom at maskinen er forbundet til flere UPSe.

UPS monitor kører på alle maskinerne, men snakker kun med UPS daemon. UPS monitor kan dog snakke med flere forskellige UPS daemons, fx hvis computeren får strøm fra flere forskellige UPSe. UPS monitor findes både til UNIX og lignende systemer,



samt også til Microsoft Windows. Muligvis kan WinNUT også være UPS driver, men jeg bruger den kun til monitoring.

Ved siden af disse 3 komponenter, så findes en del andre forskellige klienter, fx det CGI script som ses i toppen af artiklen, men også plugins til generelle service overvågnings programmer så som Nagios.

UPS driveren styres via en konfigurations fil kaldet ups.conf som kan ses her med 2 UPSe, den ene forbundet via USB og den anden via seriel.

```
Ups.conf:
[apc1500]
  driver = usbhid-ups
  port = auto
  vendorid=051d


[R3000XR]
  port = /dev/ttyS0
  driver = bcmxcp
```

UPS daemon styres via 2 konfigurations filer, upsd.conf og upsd.users. Den 1. styrer hvilke ip adresser og netværk som må tale med UPS daemon, den 2. styrer brugere, disses password og rettigheder.

```
Upsd.conf:
LISTEN 192.168.123.251
ACL all 0.0.0.0/0
ACCEPT all
```

```
upsd.users:
[upsmaster]
  password = 39w08rerhfb
  allowfrom = localhost
  upsmon master


[upsslave]
  password = elwjkdhfbedf
  allowfrom = all
  upsmon slave
```

UPS monitor styres via konfigurationsfilen upsmon.conf hvor de vigtige dele ses herunder. Pr. default indeholder upsmon.conf en del oplysninger som man ikke behøver ændre i. De vigtige oplysninger i upsmon.conf som skal ændres er:

```
# MONITOR <system> <powervalue> <password> [master|slave]
MONITOR apc1500@localhost 1 upsmaster 48eewr9 master
MONITOR R3000XR@localhost 0 upsmaster 9823wd master
MONITOR microdowell@dkserver 1 upsslave wueicx slave

# MINSUPPLIES <num> where num is how many gives power to us
MINSUPPLIES 1

NOTIFYCMD /sbin/upsmail.sh
```

I mit eksempel så får min computer godt nok power via 2 UPSe, både apc1500 og R3000XR, men den kan klare sig med strøm fra kun 1 af dem. Dvs. at systemet først lukker ned når begge UPSe er på batteri drift og når apc1500 er helt tom, da den computer er master for apc1500. For microdowell er denne maskine slave, og så stopper den før master maskinen.

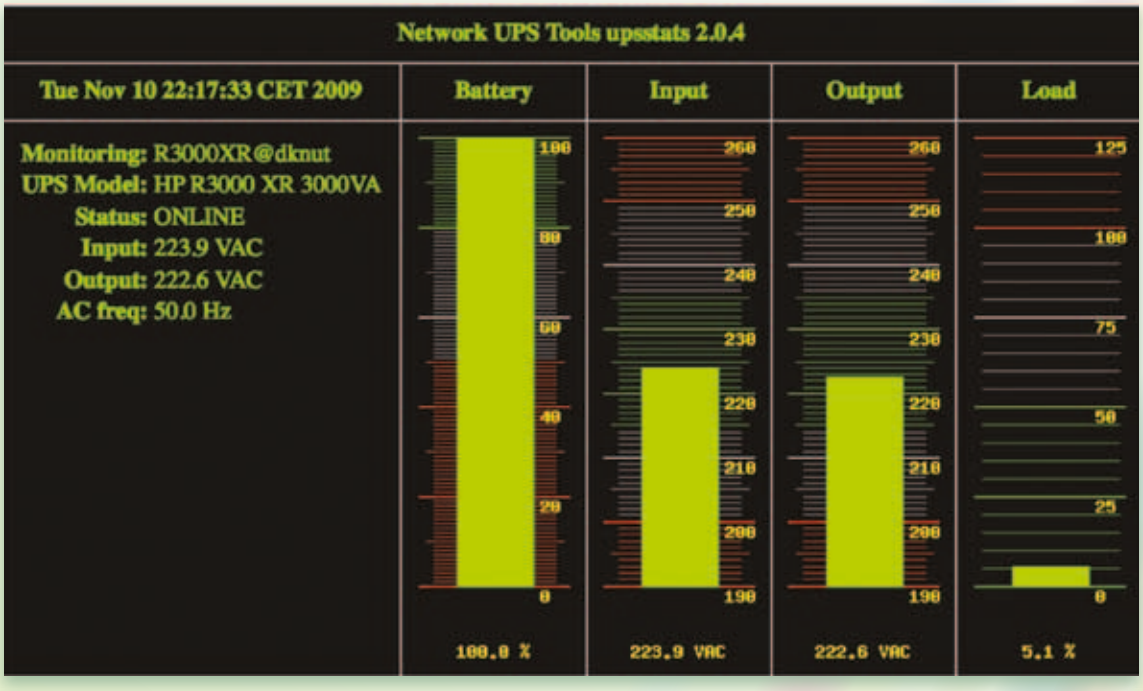
CGI scriptet som ses i begyndelsen af artiklen styres af 2 konfigurations filer, hosts.conf og upsset.conf, hvor det eneste den sidste gør er at fortælle at man har sikret sin webserver så man ikke via den kan cracke din NUT installation. Hosts.conf beskriver bare hvilke UPSe som skal vises via CGI scriptet. I mit eksempel er CGI scriptet installeret på en 3. maskine indeholder hosts.conf:

```
MONITOR match1500@dk2ovpn "Valby: GE Digital Energy Match UPS 1500"
MONITOR apc1500@dknut "APC Smart-UPS 1500"
MONITOR microdowell@dkserver "Microdowell N-30EXP 3000VA"
MONITOR R3000XR@dknut "Norways: HP R3000 XR UPS"
MONITOR bp500@dkvpn1 "Switch: Microdowell BP-500"
```

Foruden det allerede viste CGI script så indeholder pakken 2 andre CGI scripts, upsimage.cgi som hjælper med at generere grafer, og upsset.cgi som kan bruges til at ændre i settings for en UPS og sende kommandoer til den.

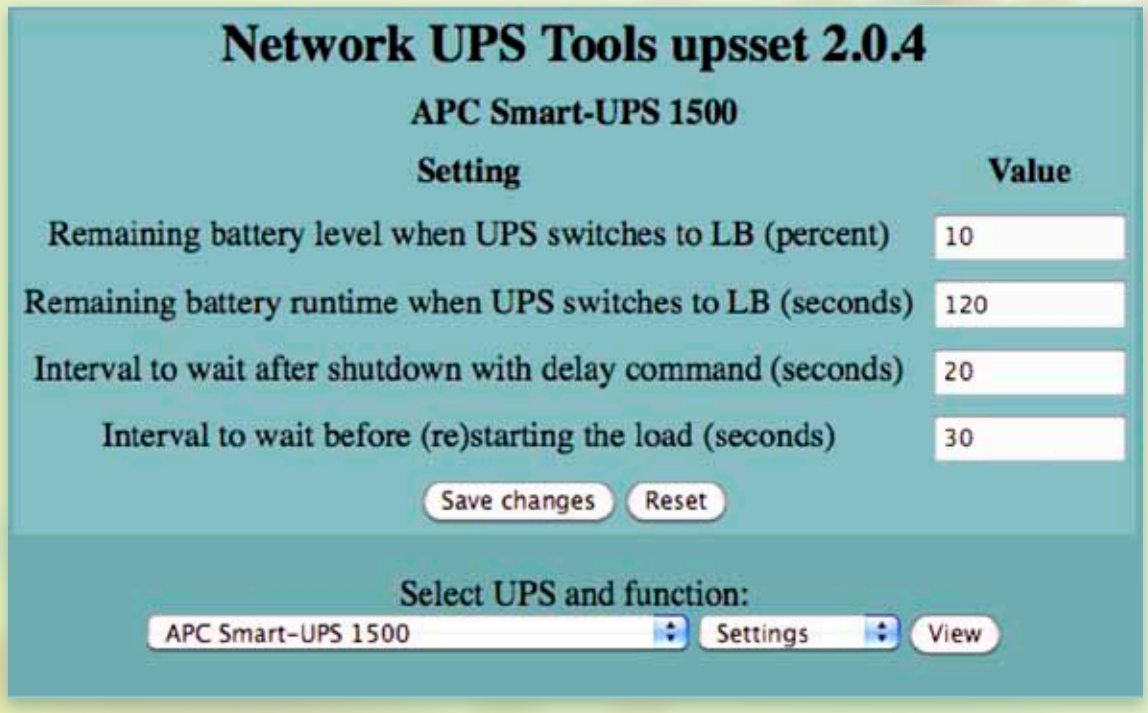
Hvis man i det viste CGI script trykker på et navn i system kolonnen, så kommer man videre ind til en detalje visning som ikke rigtigt giver nogle yderligere oplysninger.

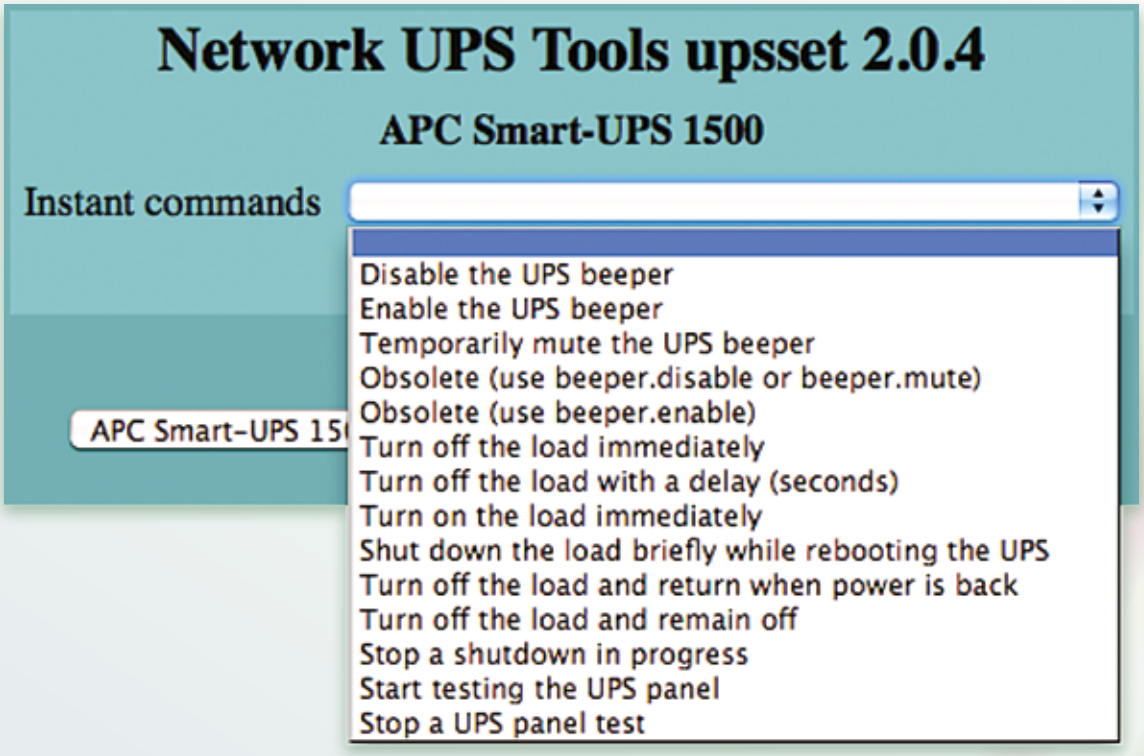
HUSK UPSen på dit netværksudstyr, ellers er NUT ikke meget værd, da forbindelserne jo går via netværk. I min opsætning er min switch UPS godt nok belastet 80%, men det er kun desktop switche og routere som sidder på den. Min server switch, hvor alt NUT trafik kører henover er på en af de andre UPSe.



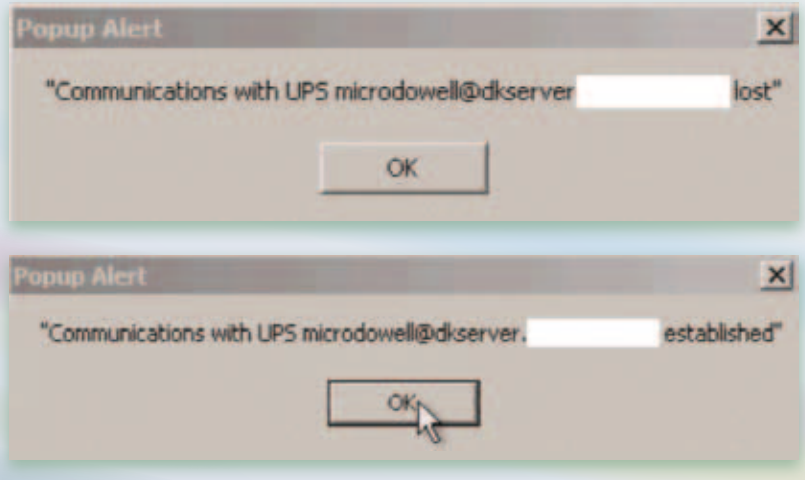
upsset.cgi giver adgang til at se og ændre i settings for en UPS, men hvilke settings som man kan se/ændre afhænger af hvilken UPS man har, samt muligvis også hvor godt UPS driveren understøtter UPSen.

Upsset.cgi giver også adgang til at sende commandoer, fx at lave en shutdown. De samme kommandoer kan man sende via en terminal, jeg har dog aldrig brugt nogle af dem. Igen afhænger kommandoerne af hvilken UPS og hvor godt UPS driveren understøtter hardwaren. Nogle fabrikanter er mere åbne end andre se http://www.networkupstools.org/acknowledgements.html



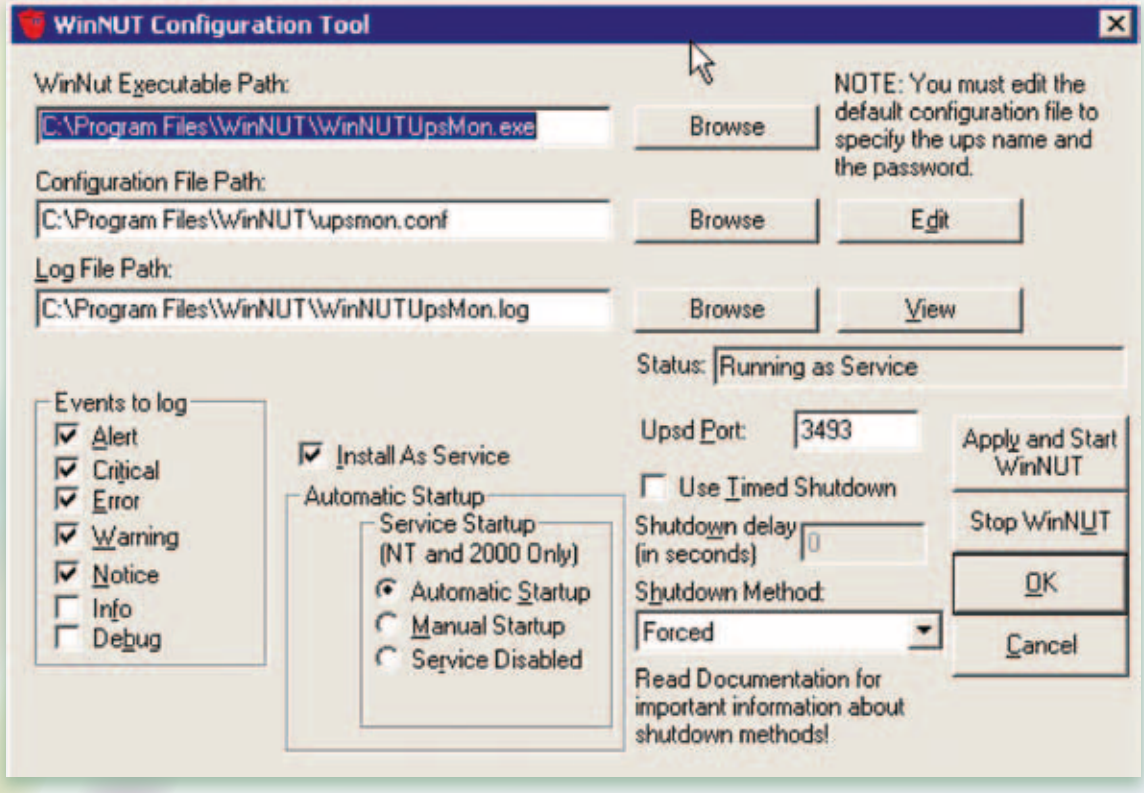


WinNUT clienten er ikke særlig smart. Den kommer pr. default med en popup når den mister eller får forbindelse til UPSen, hvilket sker for ofte, selvom at netværket stadig fungerer. Det giver efter nogle måneders drift rigtig mange popups som skal trykkes væk en af gangen. Jeg har dog endnu ikke oplevet at en windows maskine ikke lukkede ned når jeg enten bevidst har testet eller strømmen rent faktisk er gået.



WinNUT kommer med et konfigurationsværktøj som også bruges til at styre start og stop af WinNUT servicen på Windows. Dette værktøj kan også bruges til at styre mængden af popups.

Der burde være en fri åben standard for at overføre UPSens status til serveren via strømkablet, så slipper man for at trække extra kabler og have en central UPS control server. Endelig så slipper man også for manuelt at holde styr på hvilken UPS der giver strøm til en server, den slags oplysninger kunne så indsamles automatisk. Standarden burde virke således at selvom man har 2 UPSe i serie, så ville data fra den første UPS gå igennem den sidste UPS helt hen til serveren. Det skulle naturligvis også virke hvis serveren får strøm fra 2 UPSe parallelt.



Indholdet af upsmon.conf på windows er identisk med Linux udgaven.

Konklusion:

Ved at installere NUT til at overvåge dine UPSe så kan dine servere nå at lukke ned i tide således at de starter hurtigt op igen, og ikke mister data.

Har man flere UPSe er det fordelagtigt at balancere loadet således at man kan holde strøm længere. Desuden så sikrer man sig imod overbelastning af UPSen hvis strømmen går. Jeg prøver at undgå at belaste mere end 75% og helst ikke mere end 50%.

Metoden jeg bruger til balancering er simpel. Jeg forbinder en server til en UPS og skriver load tallet ned. Derefter forsøger jeg at fordele serverne hen over mine UPSe således at load bliver balanceret. Nogle servere har 2 strømforsyninger og er derfor forbundet til begge UPSe. Andre servere har kun 1 og er derfor kun forbundet til 1 UPS. Dette kan gøre det lidt besværligt at opnå samme tal i load på begge UPSe.

Da min serverpark er vokset, så har jeg været tvunget til at opgradere mine 1500VA UPSe til 3000VA UPSe. Jeg har endnu ikke nået at balancere loadet på de nye UPSe eller flytte alle servere til de nye UPSe.

Nogle gange kunne jeg godt tænke mig flere niveauer således at de servere som kan undværes lukker ned før andre, men heller ikke således at serverne lukker ned allerede når strømmen mistes, for ofte er strømafbrydelser kortvarige. Dette kunne fx være ved 80% batteri, og måske nogle andre ved 50% batteri, og så de ultravigtige som først lukker ned ved 20% batteri.

SUBVERSION –INTEGRATION

AF KENNETH GEISSHIRT - kenneth@geisshirt.dk

I de sidste 5-10 år er der sket en sand eksplosion i revisionsstyringsværktøjer. For 10 år siden var CVS og RCS toppen af poppen, men nu kan du vælge mellem Subversion, Mercurial, Git, Bazaar og en masse andre. Jeg er en glad bruger af Subversion. Ikke fordi det er bedre end de andre, men det passer godt til mine projekter. Mine projekter har typisk en masse forgreninger *(branches)* og mange udviklere. Og de udviklere, som er med, er online mens de udvikler. På den måde er Subversion et godt og moderne alternativ til CVS.

Generelt har jeg den holdning, at et revisionsstyringsværktøj ikke bør være i fokus for en gruppe udviklere. De skal jo koncentrere sig om at udvikle. Derfor er det en fordel, hvis revisionsstyringsværktøjet kan integreres med de andre værktøjer som udviklerne bruger. Her tænker jeg især på udviklernes editor.

I denne artikel vil jeg vise dig hvordan du kan integrere Subversion med en række værktøjer. Forhåbentlig vil det gøre dine udviklingsaktiviteter mere produktive.

Apache

Jeg vil begynde min integrationssnak med serveren. Det er en vigtig komponent i produktive liv for konsulent, idet jeg kommer rundt til mange klienter hvor jeg ingen indflydelse har på netværk, servere eller sikkerhedspolitik.

Det er muligt at kommunikere med Subversion over flere forskellige protokoller. Egentlig har subversion sin egen, men det er også muligt at pakke Subversion ind i en SSH-tunnel. Men det virkelig smarte – efter min mening – er at bruge HTTP eller HTTPS. Det smarte ligger i, at du aldrig er begrænset af firewalls for de vil altid tillade udadgående web-trafik. Derimod er SSH og de fleste andre porte lukkede, og du har ikke en chance for at komme i kontakt med din Subversion-server. Ved at bruge Subversion-modulet til Apache er det muligt for dig – altid – at komme i kontakt med din Subversion-server. Modulet benytter sig af WebDAV, som er en overbygning til HTTP og bruges ofte til at lade web-mastere opdatere indhold på fjerne web-server. Som Subversion opererer WebDAV med at sende forskelle mellem klienten og serveren. Det betyder at der typisk skal sende mindre end hele filer.

For at få integrationen mellem Apache og Subversion til at virke, skal du bruge Apache 2.0 eller senere. Min opsætning af Apache benytter altid SSL så jeg kommunikerer med serveren over en HTTPS-forbindelse. I nogle tilfælde køber jeg et certifikat, men i andre tilfælde bruger jeg bare et hjemmelavet certifikat. Det er lettest at bruge en *virtual host* i Apache. Nedenfor finder du en del af en Apache-konfigurationen, som giver dig en host.

```
<VirtualHost *:443>
  ServerAdmin webmaster@zigzak.net
  DocumentRoot /srv/web/zigzak.net/duck

  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>

  <Directory /srv/web/zigzak.net/duck>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
  </Directory>

  ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
  <Directory "/usr/lib/cgi-bin">
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
  </Directory>

  # SSL stuff
  SSLEngine On
  SSLCertificateFile /srv/ssl/duck.zigzak.net.pem
  SSLCertificateKeyFile /srv/ssl/duck.zigzak.net.privkey.pem
</VirtualHost>
```

Det er nu muligt at kontakte din web-server med HTTPS. Men du skal installere og aktivere Subversion-modulet til Apache. Bruger du Ubuntu eller Debian, finder du modulet i pakken *libapache2-svn*, mens FreeBSD giver dig modulet når du installerer Apache og Subversion. Du kan oprette dit *repository* på helt normal vis, men du bliver nødt til at skifte rettigheder på filer og folder for at Apache kan læse og især skrive. Med andre ord, du er nødt til at skifte ejer og gruppe så det passer med den bruger, som afvikler Apache-processerne. Udover Subversion-modulet, er du også nødt til at aktivere WebDAV-modulet i Apache.

I den oprettede *vitual host* skal du nu gøre Apache opmærksom på at den skal stå for dit *repository*. Det gør du ved at tilføje nedenstående klump konfiguration.

```
<location /svn/MitProjekt>
  DAV svn
  SVNPath /srv/svn/MitProjekt/
  # Force SSL (or https)
  SSLRequireSSL
  # Authentication
  AuthType Basic
  AuthName "mitprojekt"
  AuthUserFile /srv/svn/mitprojekt.passwd
  require valid-user
  Order deny,allow
</Location>
```

I eksemplet finder du et *repository* ved navn **MitProjekt.** Som du kan se, kræver det et gyldigt brugernavn og adgangskode for at læse og skrive i **MitProjekt.** Du kan bruge alle former for autentikering som Apache understøtter – dog har jeg i Ubuntu Server 8.04 (LTS) set nogle problemer med at bruge alm. htpasswd. Oftere bruger jeg PAM eller nogle af de nyere mekanismer.

Med dit repository på plads i Apache, kan du uden problemer udføre en *check out*. Gør du det fra en kommando-linje er kommandoen **svn check-out --username kneth https://duck.zigzak.net/svn/MitProjekt.** Du er nødt til at angive brugernavnet, men så vil Subversion-klienten spørge dig om adgangskode. Når du har givet brugernavnet og adgangskoden første gang, vil du ikke blive spurgt mere om det.
Sidder du bag en proxy-server, kan Subversion-klienten også klare det. I konfigurationsfilen **.subversion/servers** kan du angive brugen af en proxy-server.

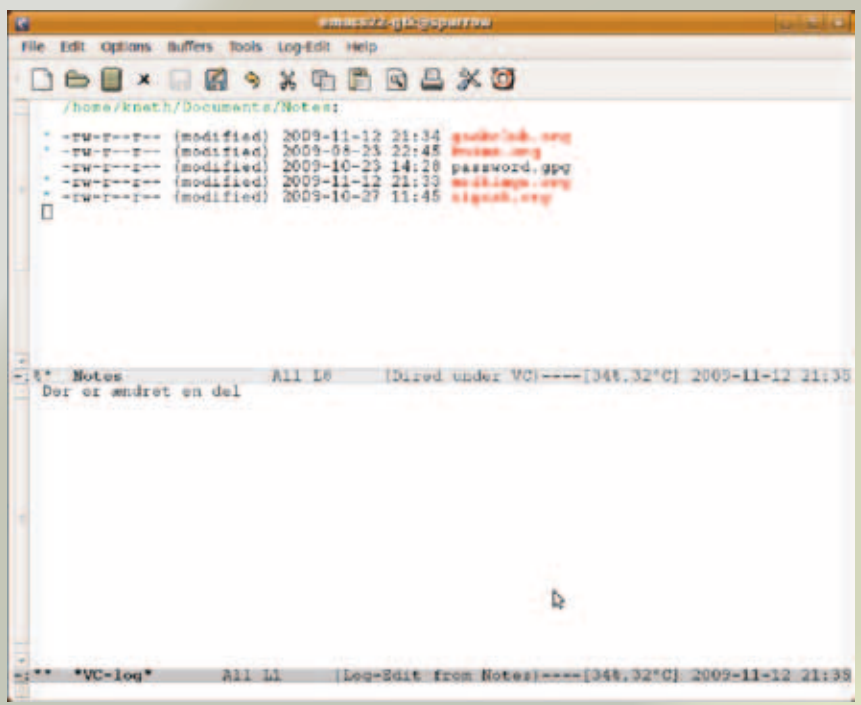
Emacs

Lad mig indrømme det med det samme – jeg er Emacs-bruger. I min dagligdag lever jeg meget af min tid i Emacs. Udover kildetekst, bruger jeg Emacs til notater, timeregistrering og regnskab. Alt i alt forsøger jeg at holde mit liv i klar tekst.

Heldigvis er Emacs veludrustet til håndtering af revisionsstyring. I de sidste par større versioner af GNU Emacs (22 og 23) har der fulgt VC med. VC står for *Version Control* og er et *minor mode* til Emacs. Oprindelig understøttede VC kun RCS og CVS, men i de senere par år er kredsen af understøttede revisionsstyringsværktøjer udvidet kraftigt. Subversion er ingen undtagelse. Med VC behøver jeg meget sjældent at bruge en kommando-linje til at arbejde med Subversion. Jeg kan klare det meste fra Emacs mens jeg redigerer filen. VC vil automatisk se at en fil er under Subversions kontrol og du behøver ikke at gøre noget særligt for at få Emacs til bruge VC.

VC er bygget op omkring konceptet *næste naturlige handling*. Ideen bag konceptet er at der er en naturlig arbejdsgang i din omgang med dine filer. Du kan godt vælge noget andet end den næste naturlige handling, men i over 90 pct. af alle tilfælde, rammer VC rigtigt. For eksempel er næste handling at tilføje filen, hvis filen ikke er tilføjet før og næste naturlige handling for en ændret fil er at sende ændringen til Subversion-serveren. Alt i Emacs er bundet til kombinationer af tastetryk, og næste naturlige handling er **Ctrl-x v v.**

Har du ændret i mange filer og udgør ændringer et samlet logisk hele, er det en fordel at *commit* ændringer sammen. Derved får de samme revisionsnummer og samme ledetekst. Med VC kan du med **Ctrl-x v d** få VC til at finde alle filer, som er ændret i en given folder (og alle dens underfoldere). I skærmbilledet nedenfor ser du hvordan det ser ud i Emacs.



Og de andre

Det er ikke kun Emacs, som har en fin integration med Subversion. Både GNOME, KDE, Windows og Mac OS X har klienter eller plugins til deres filbrowsere som gør det muligt at arbejde med Subversion.

Vim er en meget populær udgave af den klassiske editor vi. Der findes et plugin til Vim, som giver adgang til Subversion. På samme måde findes der til Eclipse mulighed for at installere et plugin til håndtering af Subversion – navnet på dette plugin er Subclipse. Eftersom Eclipse er et meget grafisk udviklingsmiljø, giver Subclipse dig mulighed for at se historikken af dine projekters filer som finde grafer (DAGs).

Der findes en extension til OpenOffice.org som lader dig gemme dine dokumenter i et Subversion-repository, jeg har dog endnu ikke haft tid til at afprøve det. Måske der engang kommer en fremtidig artikel omkring det.

Få mere at vide

- http://subversion.tigris.org/
  - Subversions hjemmeside
- http://www.emacswiki.org/emacs/VersionControl
  – Kort introduktion til VC
- http://www.vim.org/scripts/script.php?script_id=922
  - Vim/Subversion-integration
- http://subclipse.tigris.org/
  - Eclipse/Subversion-integration
- http://extensions.services.openoffice.org/project/OOoSVN
  - OpenOffice/Subversion-integration

Afslutning

Jeg håber at du med denne artikel har fået et indtryk af at Subversion kan integreres med en lang række andre open source applikationer. Langt de fleste applikationer er nok rettet mod softwareudviklere, men som eksemplet med OpenOffice.org viser, kan Subversion også bruges i andre sammenhænge.

Har du lyst til at skrive en artikel?
kontakt bld@dkuug.dk

Har du lyst til at holde et foredrag?
kontakt klb@dkuug.dk

Har du lyst til at optage video til et arrangement? kontakt video@dkuug.dk