DKuug nr. 173

Januar-februar 2014

NYT

Symbion på Fruebjergvej ved Lersø Parkallé og Ryparken, vores ramme om IT arrangementer

Dansk Forum for Åbne Systemer

DKUUG - Unix Brugere (Linux/BSD) system administratorer
Community for IT-specialister og IT-interesserede.

DKUUG

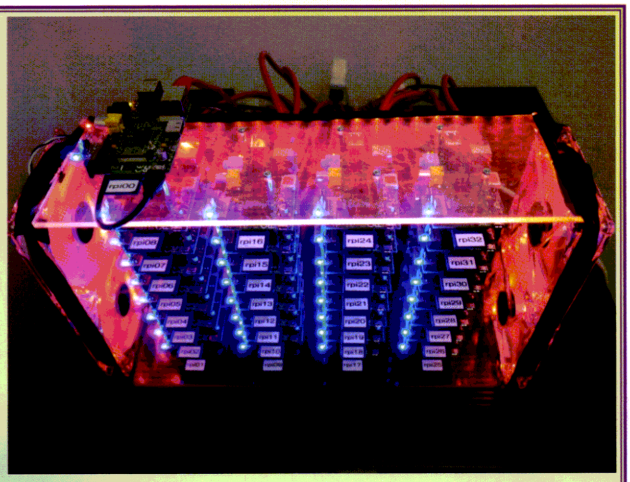
I dette nummer

viser David Askirk Fotel en applikation til Raspberry Pi. Det er en singleboard computer, som kan fås for lidt mindre end en DSB returbillet til Odense koster, ca. 300 kr og ca. 400 kr hvis man vil have en 8 GB solid-state disk med boot-image med i købet. Det er så billigt, at man fristes til at prøve. Rundt om i verden sidder ingeniører og amatører og bygger systemer med Raspberry. Som fx. Joshua Kiepert, Ph.D studerende ved Boise State's Electrical and Computer Engineering department i Idaho USA, der med hjælp fra andre studerende og lærere har bygget et cluster af RaspberryPi boards. Se bare --->

Kiepert har målt 10.13 GFLOPS (med Linpack benchmark) og er selvfølgelig klar over at det ikke giver en plads på super-computernes TOP500.

Den første Cray-2 i 1985 ydede 1.9 GFLOPS - og 10 GFLOPS er rigeligt til at løse realistiske opgaver og forstå, hvordan parallel computing fungerer.

Undertegnede viser som lovet i sidste nummer en web-applikation, der læser og skriver i en database. Det har kostet tid at finde en balance mellem realisme og overskuelighed, og det er endt med vægt på forsimpling. Det var desværre nød-vendigt at forudsætte, at læseren har lidt kendskab til HTML *tags*.



Video-artikel

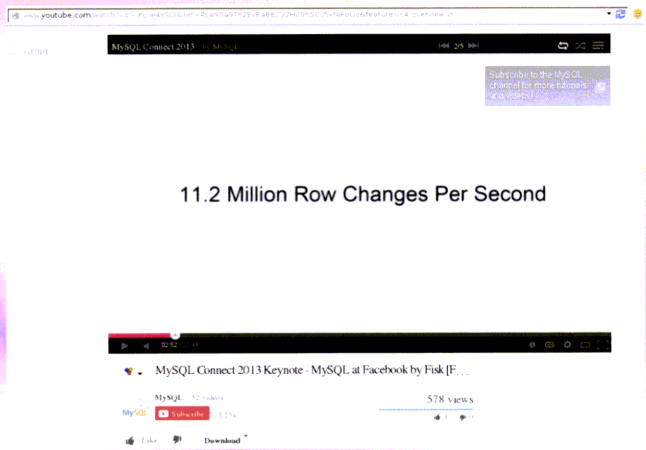
Jon Bendtsen beskriver i en artikel hvordan DKuugs video-gruppe har opbygget et system af scripts, som sætter en forside og logo på de video'er, som er optaget på OSD-2013 og ved foredrag. De ligger på video.dkuug.dk

Donald Axel

Database-artiklen i sidste nummer

handlede om PostgreSQL, som performancemæssigt er kommet op på siden af MySQL.

Desværre fik vi ikke nævnt, at en MySQL database med samme 6 mio. records var nøjagtig lige så hurtig som Postgres, måske en anelse bedre. - Det er hermed slået fast! MySQL er uhyre udbredt fordi den i de tidlige web-tider kunne levere høj



performance.

Har du nogen gange tænkt på, hvor meget MySQL kan trække? Har du tænkt over, hvad der sker, hver gang du klikker "like" på et indlæg eller en kommentar på Facebook? Facebook bruger MySQL.

I en video på YouTube fortæller Harrison Fisk fra Facebook om MySQL, Hadoop og omkringliggende applikationer.

Facebook kan forbinde kontinenter, men serverdata ligger spredt på servere over hele klodens civiliserede områder. De samlede tal skal derfor ikke tages som udtryk for en vidunderligt hurtig maskines performance, selvfølgelig, men som udtryk for, at MySQL kan være rygraden i et cluster af meget aktive systemer. Som facebook bruger har jeg prøvet at skabe nogle kontakter fra DK til USA og flere andre steder og kan konstatere, at det ikke altid kører lige hurtigt. Der er ligesom landegrænser indbygget - men det kan lade sig gøre at have forbindelser i USA og Korea m.v.

Facebook performance er interessant - se videoen, søg på *MySQL Connect 2013 Keynote - MySQL at Facebook by Fisk*.

60 millioner select pr sek.! 42 mio. IO, 24 mio disk IO pr.sec, men kun 0.4 disk-IO pr. query idet man har mange forskellige caching systemer hos Facebook.

Somme tider overvejer vi i redaktionen om DNyt er lidt for nørdet. Der er læsere, som godt kan lide artikler, der viser de problemer en systemadministrator har, klart nok, men vi bliver færre og færre. Bliver nye læsere skræmt væk? Vi vil i de kommende numre prøve at se mere på de trends, som overordnet set skaber interesse for systemadministration på Unix-systemer og vi håber at skabe et blad med både detaljer og overblik.

Det er fordi vi mener at en programmørs viden er grundlæggende funktionsmekanismer i IT, og detaljer, kommandoer, vinduer, knappenavne, mv. varierer fra år til år, fra produkt til andet produkt og fra release til release. Derfor er det vigtigere for IT-professionelle at kunne navigere i viden - mon ikke IT-administratorer er nogle af dem, som har mest glæde af Google? Tænk på alle de gange, man har fundet svar på et problem i et forum - stackoverflow fx. - via en søgning på Google.

Næste nummer

Engang for mange år siden hørte jeg den ansvarlige chef i en TDC afdeling sige til direktøren, at han ikke endnu havde nået at sætte sig ind i alle de filer, der skulle være på den AIX-Unix-opdatering, som de brugte. Det ville man slet ikke forsøge på i dag, men omvendt er det nødvendigt at holde styr på om en maskine er kompromitteret eller ej. Derfor må man bruge check-systemer.

Mængden af filer på selv en lille workstation skal tælles i tusinder. På godt og ondt. På de første systemer med floppy-diskette, som min generation oplevede fra ca. 1980, var mængden af filer sjældent over 100. Læsernes kommentarer er velkomne på blad@dkuug.dk

Vi vil fortsætte serien om Open Source databaser, i næste nummer MySQL.

Donald Axel

DKuug-NYT er medlemsblad for DKuug, foreningen for Åbne Systemer og Internet
Nr. 173 - Januar 2014

Udgiver:
DKUUG
Fruebjergvej 3
2100 København Ø
Tlf. 39 17 99 44
email: dkuugnyt@dkuug.dk
Redaktion:
Donald Axel (ansvarshavende)

Forsidecredits:
(redaktionen)

Design og layout:
DKUUG/Donald Axel
Annoncer:
pr@dkuug.dk
Tryk:
Lasertryk i Aarhus
Oplag:
500 eksemplarer

Artikler og inlæg i DKUUG-Nyt er ikke nødvendigvis i overensstemmelse med redaktionens eller DKUUGs bestyrelses synspunkter.

Eftertryk i uddrag med kildeangivelse er tilladt.

Deadline for nr. 174: 9. Marts 2014

Medlem af Dansk Fagpresse
DKUUG-Nyt
ISSN-1395-1440

Vores møder og **foredrag** holdes - med mindre andet udtrykkeligt angives - på vores adresse:

DKUUG
SYMBION
Fruebjergvej 3
2100 København Ø

Hvis man kommer lidt før, er der tid til en snak på kontoret. DKUUG bor i en virksomhedsfarm, Symbion, hvor der er åbne døre indtil kl.18. Efter den tid har vi på foredragsaftener en vagt ved døren.

INDHOLD:

Arrangementer:

DKUUG åbent hus onsdag d. 12 marts 2014 på vores kontor - se hvor du selv kan arrangere kurser i Symbion.

Bryan Østergaard
Visual editor - improved: **Vim** og de grundlæggende fordele ved **vim.**
d. 19 Februar - kl.19.
Bent Bagger
Linux: Marts i Symbion - kl. 19. Nærmere annoncering via web og mail.

OSD-2014: Forventes i slutningen af april.

Andre arrangementer vil blive annonceret på web og via mail.

Gør-det-selv foredrag:

Kom og få dine kompetencer plejet - hold et foredrag eller workshop om det, der interesserer dig.

Vores kontor i Symbion giver mulighed for hands-on workshops, både dag og aften. Skriv til pr@dkuug.dk eller til bestyrelsen i DKUUG, bestyr@dkuug.dk, og hør om lokalet er ledigt den dag du vil arrangere et møde. Der er hurtig internetforbindelse, både wired og wireless. Er der større tilmelding, kan vi leje mødelokaler i Symbions mødested. Spørg os!

Deadline for DNyt nr. 174: Søndag d. 9 marts 2014.

Raspberry Pi Post sensor

af David Askirk Fotel

Mange har i dag hørt om Raspberry Pi computeren. Den lille, billige ARM baserede computer som man kan bruge til (næsten) alt.

Men hvordan gør man? Hvordan bruger man den?

I denne artikel laver vi en postkasse sensor, så vi ikke behøver at gå ud til vores postkasse for at se om der er kommet post. Vi putter også et web-interface på så vi ikke engang behøver at forlade vores computer.

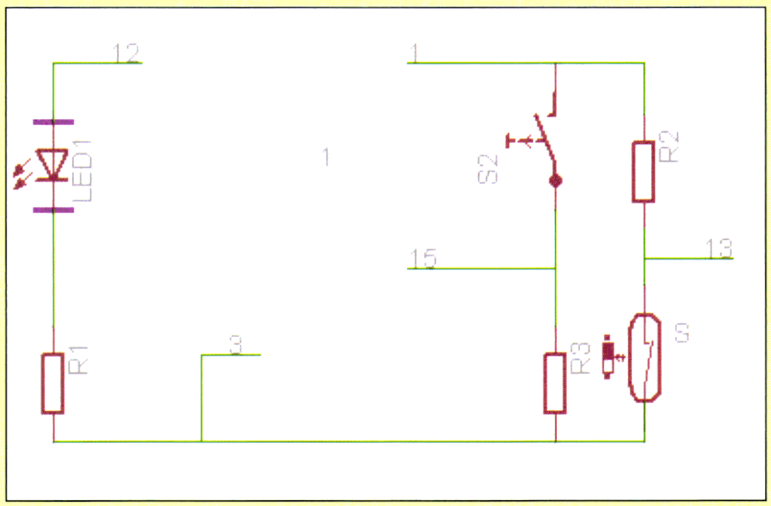
Vi skal bruge nogle forskellige elektroniske komponenter:

- 1 Raspberry Pi
- 1 Reed kontakt med 1 lille magnet (S1)
- 1 Alm. tryk kontakt (S2)
- 3 modstande (R1-R3)
- 1 LED (LED1)
- Ledninger og evt. breadboard eller noget andet prototyping print

En ***reed*** kontakt er en kontakt, der reagerer på en magnet.

Hvis man bare vil lege, kan man benytte en alm. tryk kontakt istedet for en reed kontakt.

Det elektroniske diagram ser sådan ud:



De små tal på enderne angiver på hvilken pin på raspberry pi'en de skal forbindes til.

Man kan se hvad de forskellige ben på en pi rev. 1. betyder her:

http://raspberrypi.znix.com/hipidocs/topic_gpiopins_rev1.htm

Men elektronik kan ikke gøre det alene. Der skal også skrives noget software til at styre det hele. Softwaren bliver skrevet i python. Programmet følger her:

```
import RPi.GPIO as GPIO
from bottle import route, run

LED_pin = 12
reset_pin = 15
mail_pin = 13
mail_is_present = False

def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(reset_pin, GPIO.IN)
        GPIO.setup(mail_pin, GPIO.IN)
        GPIO.setup(LED_pin, GPIO.OUT)

def handle_mail(chan):
        global mail_is_present
        mail_is_present = True
        GPIO.output(LED_pin, GPIO.HIGH)

def handle_reset(chan):
        global mail_is_present
        mail_is_present = False
        GPIO.output(LED_pin, GPIO.LOW)
```

```
setup()

GPIO.add_event_detect(mail_pin, GPIO.RISING, callback=handle_mail, bouncetime=200)
GPIO.add_event_detect(reset_pin, GPIO.RISING, callback=handle_reset, bouncetime=200)

@route('/')
def index():
        return '{"mail":'+str(mail_is_present)+'}'

@route('/reset')
def reset():
        handle_reset(None)
        return 'OK'
run(host='0.0.0.0', port='8090')
```

Hvis man installerer Raspbian (Debian for raspberry pi) så følger der automatisk GPIO biblioteket med. Det er det vi benytter os af til at snakke med knapperne og LED'en.

Programmet kan køres i en screen, så det kan køre selvom man er logget af systemet.

Bottle.py bliver brugt til at serve et webinterface til brugeren.

I setup metoden bliver de forskellige pins sat op, og gjort klar til brug. Dem, hvor kontakterne sidder bliver erklæret som input og der hvor LED'en sidder, som output.

Derefter bliver der erklæret to metoder. De bruges til at håndtere et tryk på hver kontakt.

Disse bliver bundet til pins med denne kommando:

```
GPIO.add_event_detect(reset_pin, GPIO.RISING, callback=handle_reset, bouncetime=200)
```

Den kigger på hvornår der er en rising edge, altså hvornår der sker en ændring. Den benytter sig også af software debouncing. Dette betyder at hvis man trykker på kontakten bliver det næste tryk inden for 200 ms ignoreret. Dette gøres for at undgå at kontakten står og hopper, som kontakter godt kan når man trykker på dem.

Hvis man så lige vil teste kan man gå ind på:

http://192.168.0.8:8090/ (192.168.0.8 er min Raspberry Pi's IP, hvis man ikke lige kan finde den kan man lige køre en sudo nmap -O 192.168.0.2-50 i mit tilfælde, eller tilslutte en skærm til den.)

Efter at have besøgt adressen med sin browser får man følgende svar:

```
{"mail":False}
```

Dette fortæller at der ikke er kommet noget post. Det er i JSON format lige til at putte ind i sin nodejs applikation eller noget andet der benytter sig af JSON til kommunikering.

Det andet kald man kan lave resetter systemet. Det kan kaldes når man har hentet posten, så den er klar til næste gang der kommer post.

Når man nu har sat en raspberry pi op til at overvåge ens postkasse, kan man udbygge den. fx med et raspberry pi cam, eller et alm. usb webcamera. Så kan man bygge et overvågnings system ind i dette. Man kunne også forbinde ens ringklokke til sin pi, ved at bruge det fra denne artikel og på den måde gemme hvornår folk ringer på ens dør.

Raspberry Pi'en er en lille arm computer, som man kan bruge til mange ting, ikke blot postkasse sensor. Den kører en linux så man kan installere alt hvad man har lyst til at køre på den, som fx. ens irc klient, eller en minecraft server.

Links:

http://code.google.com/p/raspberry-gpio-python/ - GPIO python biblioteket som bliver brugt.

http://www.raspberrypi.org/ - Raspberry Pi foundation. Dem der har lavet Raspberry Pi.

http://bottlepy.org/docs/dev/ - Bottle.py, det micro web-framework der bliver benyttet i denne artikel.

Raspberry-Pi

Single board computer for fremme af grundlæggende datalogi

Over 2mio Raspberry Pi (pai som i apple-pie) er solgt for ca. 280 kr. stykket siden begyndelsen i februar 2012.

Der tog kun et år at få solgt den første million, men salget steg, og anden million blev nået i slutningen af oktober 2013.

"Det tog næsten nøjagtigt et år at sælge den første million", skrev Liz Upton fra Raspberry Pi Foundation. "Ud fra dette håbede vi at anden million kunne være solgt omkring januar 2014, eller lidt efter, men vi var ret sikre på at vi ville nå dertil i slutningen af februar 2014."

Hun fortsætter: "Det var lidt af et shock, da vi fik salgstallene i oktober og opdagede, at Raspberry Pi nr. 2,000,000 var solgt i slutningen af oktober."

Velgørende organisation

Raspberry Pi Foundation er en non-profit organisation, som er dannet i 2009, hvis formål er at fremme studiet af computer-science (datalogi) og lignende emner på et niveau, så skolebørn kan være med - men ingenlunde begrænset til skoleniveau.

Det er en reaktion på, at så mange børn bliver skoletrætte efter 7-8 år i skole og mister lysten til at lære. Med Raspberry Pi kan man - forhåbentlig - få morskaben tilbage i indlæringen. Antallet af solgte eksemplarer tyder på det.

Eben Upton, som var med til at grundlægge Raspberry Pi, har en akademisk karriere bag sig men er nu ansat hos broadcom som arkitekt for system-på-en chip projekter og er associeret teknisk direktør.

Ideen var at bruge et simpelt programmeringssprog, *scratch,* og dertil have externe enheder, som computeren kunne kontrollere.

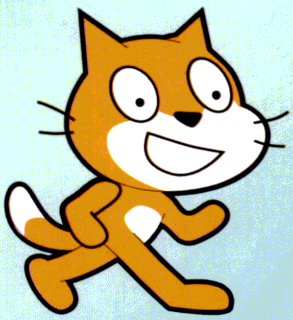


Illustration 1: Scratch er et programmeringssprog til undervisningsbrug og til multimedia programmering, med event-drevne multiple aktive elementer, som kaldes sprites

Med andre ord taler vi ikke om programmering af apps til Apple- iPhone eller smartphones, men om et grundlæggende system, som kan programmeres på flere måder, som kan udvides hardwaremæssigt, og som derfor kan vise alle grundlæggende datalogiske discipliner.

Man lærer forbindelsen mellem elektronik og mekanik på et plan, som er til at finde ud af for begyndere. Programmering af selv de mindste ting til smartphones kræver som brug af biblioteker, dvs. hjælperutiner til styring af grafisk user interface, GUI, der er samlet i library-filer eller biblioteksfiler, og det behøver vi vist ikke at gå nærmere ind på her (se Davids artikel).

Når man programmerer Raspberry Pi er man tættere på den grundlæggende funktionalitet i elektronikken - på samme måde, som fx. termostater kunne styres med Intel 8051 som controller (i dag opdateret til 8/16/32 bit binær kompatibel MCS-251).

Men til forskel fra Intel-MCS251 er Raspberry Pi udstyret med en ægte RISC processor, ARM, Acorn Risc Machine: Acorn er holdet af ingeniører, som i 1979 lavede en hjemmecomputer, som BBC brugte i skole-udsendelser, deraf navnet BBC-computeren.

Det er på denne baggrund forståeligt, at Raspberry Pi ud over programmeringssproget *Python* kører *BBC-Basic,* en Basic dialekt med ekstra struktur-keywords: Man kan definere procedurer og funktioner. Men på Raspberry Pi er det meningen at Python skal være det mest almindelige programmeringssprog.

ARM chippen er kendt for at være hurtig og bruges i mange smartphones. Linux kører fint på ARM.

Raspberry Pi specifikationer

Den oprindelige Raspberry Pi kostede 25 $ og havde 256 MB memory. Den senere model B har 512 MB RAM og koster 35 $.

Operativsystemet er en Linux-kerne konfigureret og kaldt Raspbian afledt af Debian Linux, (men man kan også køre andre Linux distributioner: Fedora, Arch Linux ARM, Gentoo) RISC-OS, FreeBSD, NetBSD, Plan 9, Openwrt. Model A bruger 2.5 Watt, B (512 MB) bruger 3.5 Watt.

CPUen er en ARM1176JZF-S (ARMv6k) 700 MHz.

Den har ikke indbygget harddisk eller solid state disk (SD-disk) men bruger et SD-kort til at boote og som permanent lager (disk).

Den kan godt tage SDHC kort.

Grafik-processor

Desuden har den grafik onboard, Broadcom VideoCore IV - som deler RAM med CPU'en. GPU'en nås gennem kode, som loades i GPUen fra SD-kort. Denne kode er en slags firmware kode, en binær "BLOB" (Binary Large Object). De tilhørende oprindelige Linux-drivere til denne GPU var oprindeligt closed-source, men nu er de released som open source - men kun den del, der kommunikerer med BLOB-driveren, som det kan forstås ud af diagrammet:

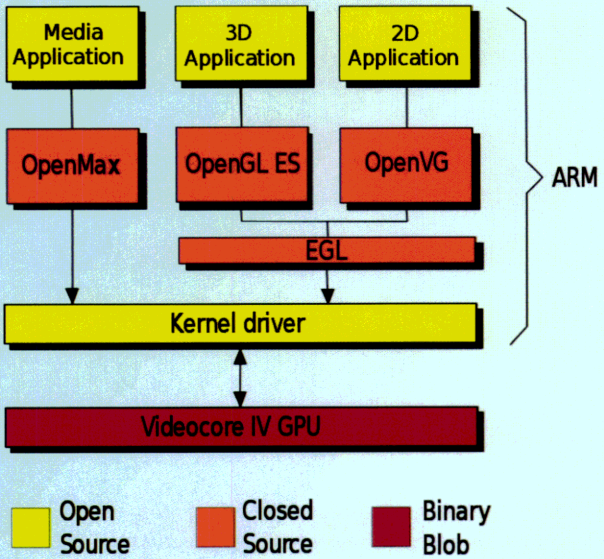


Illustration 2: Closed/open source elementer i Raspberry Pi grafiksystemet - med tak til Wikipedia/Wirepath

Se mere på *www.raspberrypi.org.*

Strømforsyning

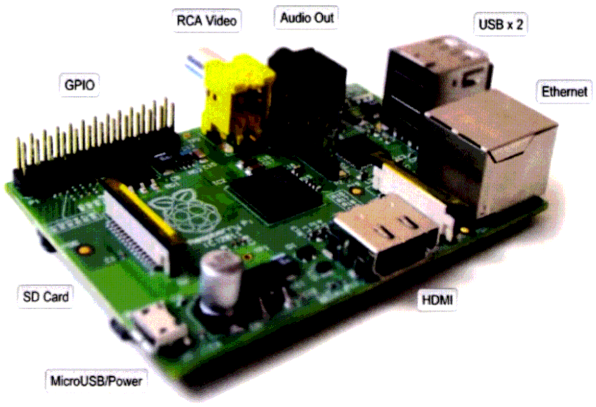
Raspberry Pi får strømforsyning gennem et USB kabel. Den bruger 5 Volt - min. 4.75, max 5.2 V 700 mA, men hvis man sætter tastatur og mus på skal den have mere, 1000 mA, og mere er bedre. Det anbefales at man bruger en USB

Raspberry Pi - Model B Specs

CPU: ARM1176JZF-S 700 MHz
GPU: VideoCore IV
RAM: 256 megabytes

Ports

USB 2.0 x 2
HDMI
RCA Video Out
Stereo Audio Out
Ethernet 10/100
GPIO Header
Micro USB (Power Only) 5v
SD/SDHC Slot



strømforsyning. Hvis man ikke har tænkt sig at den skal køre i døgndrift kan man bruge USB kabel fra en desktop computer.

Men man kan risikere at den er ustabil hvis man giver for lidt strøm. En mobil-oplader giver typisk ikke så mange mA som den er specificeret til, og så kan man risikere at Raspberry Pi crasher.

TechErudio tutor video på YouTube fortæller, at selv med en 2000 mA USB-strømforsyning fik han stadig ikke en stabil maskine. Han viser i tutor-video nr.2 hvordan han måler spændingen med et (billigt) voltmeter, og afhængig af hvad han kørte på maskinen viste det sig at spændingen faldt under 4.75 V hvorfor systemet bliver ustabilt. Men interessant er det, at TechErudio finder ud af at det ikke er selve strømforsyningen, men kablet, USB-kablet, som bevirker at Raspberry'en ikke fik nok strøm. Forklaringen kan være længden af kablet eller andre egenskaber ved kablet.

Hovedløs konfiguration

Nu er det så allerede afsløret, at man kan køre med forskellige konfigurationer. Men det væsentlige er egentlig at man kan køre uden tastatur og keyboard - headless.

Hvis man kører som headless server, kan man nøjes med at låne et tastatur og skærm fra en workstation til det første setup efter installation af operativsystem.

Men hvis man har planer om at arbejde lokalt (udenom netværk) regelmæssigt med Raspberry skal man bruge:

- 1 SD kort (eller SDHC)
- 1 USB tastatur, USB mus,
- 1 netkabel kategori 5 eller 6 (man kan dog også bruge trådløs ved anskaffelse af ekstra extern device)
- Strømforsyning og god USB ledning til denne
- Router (eller ekstra netkort i en router-konfigureret Linux Workstation)
- Skærm og monitorkabel: HDMI eller HDMI til DVI (afhængig af skærmens stik) eller en adapter (i så tilfælde er der ikke lyd gennem video-stikket).

Der er også et RCA video-output stik, som fungerer, dog ikke til HD videoer. TechErudio anbefaler også at man anskaffer et kabinet, som koster ca. 50 kr.

Anvendelser

Raspberry Pi kan anvendes til kontroller, til router, til grafisk rendering (d.v.s. beregningstunge opgaver, evt. i cluster). Der er også mulighed for at anvende den som media center (XBMC) takket være grafikenheden. Der bliver lagt mange kræfter i at forbedre XBMC performance, og til udviklernes forundring har man set at Raspberry nu har flere XBMX-brugere end Apple TV2. (***http://www.raspberrypi.org/archives/4986***)

Raspberry Pi kan købes herhjemme fra raspberrypi.dk og elav.dk, som også har samlede pakker. Fx.raspberrypi.dk model B med 8GB SD-disk med image (til boot) 399 kr. eller bare board model B fra elav til 299 kr. En samlet XBMC tilbehørs-pakke koster 649 kr hos raspberrypi.dk

Boardet fås også fx. hos Proshop DK. Her er prisen for model B 349 kr. incl. moms. I skrivende stund melder forretningen, at der er nogle stykker på lager:

15+ stk på lager i Århus
4 stk på lager i Viborg

Radio Shack (RS) er jo ellers en leverandør, som sælger alle mulige komponener - og hvis man bruger den tyske afdeling, burde pakkerne gå glat igennem. EU handelsrestriktioner og privat handel er et kapitel for sig, men det bør lade sig gøre at købe via de.rs-online.com

En aftale mellem RS og Raspberry Pi Foundation har gjort det muligt at fabrikationen foregår i en Sony virksomhed i Wales.

Ikke alle mener at Raspberry Pi kan kurere skoletræthed: Skolernes netværk og andet udstyr i UK er 20 år gammelt og der venter mere arbejde før skolerne er parate til Pi, forklarer Nick Williams, som er produkt-chef hos Brocade Communications, UK.

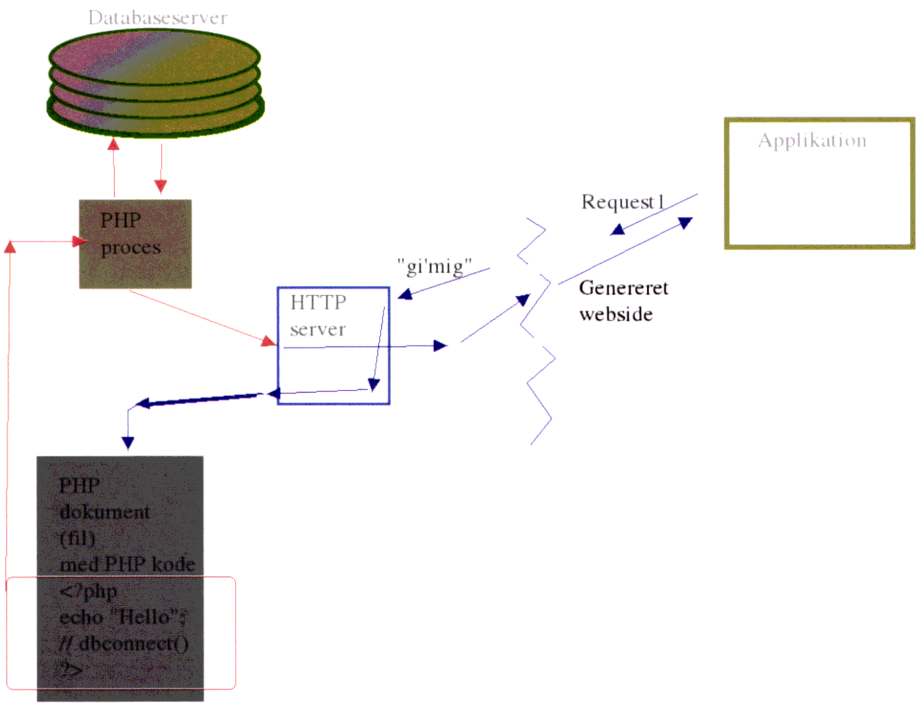
Dynamisk webapplikation

En minimal web-applikation giver adgang til opdatering af database

Af Donald Axel

I forlængelse af PostgreSQL artiklen i sidste nummer af DNyt giver vi her et eksempel på en lille bitte minimalistisk applikation, som kan køre på en hvilken som helst browser. Dermed åbner man for anvendelse på en smartphone. Ved passende brug af java-script knapper vil man kunne få applikationen til at fungere med tommelfingertryk på en smartphone.

Lad os stille en forsimplet opgave: Fjernt fra hjemmets trygge internetforbindelse registreres observationer af fuglearter, en artsoptælling som dem, Dansk Ornitologisk Forening foretager for at få en ide om bestandenes størrelse. Men det er ikke opgaven, som er vigtig, det er måden systemet fungerer. Her er et skematisk overblik over http-server, CGI-script og database:



Fred Grott specialiserer sig i udvikling af smartphone applikationer og har to forretningsmodeller: Han kan udvikle en specialiseret applikation i native code som udnytter alle muligheder i en smartphone, eller han kan skrive en web-applikation, som fungerer i alle webbrowsere på alle almindelige nyere systemer.

Gæt hvilken af disse forretningsmodeller, som som kræver 85,000 kr investering og hvilken der kræver 1500,000 kr? Ud fra økonomiske betragtninger er Grotts valg simpelt: Tag en almindelig web-applikation og tilføj logik til at bestemme hvilken browser vi kører på, send en lidt tilpasset webside tilbage, tilpasset skærmen på mindre devices.

Udviklere på smartphone markedet bruger forskellige modeller, men en af de mest oversete udviklingsmodeller er den tilpassede web-applikation, skev ***infoworld*** i januar 2009.

Til en begyndelse skal der registreres en artsbetegnelse og antal. Stedet kan være konstant eller hentes fra en GPS - foreløbig lader vi det være. Et nummer eller en knap kan være et shortcut, hvorved indtastning holdes på et minimum - og antal kan også kan have shortcut-knapper (efter princippet 1, 2, mange) - derved vil en webapp på smartphone have en funktionalitet, der matcher en "rigtig" app. Et realistisk design ville med javascript opsamle et batch af observationer og sende dem samlet til basen.

Design

For at se hvordan vores web-applikation vil fungere, hvis vi lader databaseserveren køre på en dertil optimeret maskine og http-serveren på en front-maskine, lader vi database-serveren lytte på en TCP-socket.

Hvis den endelige løsning bliver opsat sådan, at database-serveren kører på en back-end på et internt net, vil database-maskinen ikke være synlig udefra. Det kan være en fordel.

Man kan man godt nøjes med én maskine, hvorpå der opsættes en apache-server og en PostgreSQL database. Og man behøver selvfølgelig ikke at lade databasen "snakke" over port 5432, men hvis en webservice - med komplicerede websider - kommer op over 400 hits i minuttet vil det være nødvendigt at fordele belastningen på flere maskiner.

Apache HTTPD - web serveren

Apache webserverens historie beskrives i korte træk på httpd.apache.org. Andre dele af historien er spredt på nettet og i bøger, Berners Lee's bog om Internettets baggrund nævner den, andre steder kan man læse om hvor forbavsede IBM var, da de opdagede at man ikke kunne købe Apache. Navnet er et ordspil på "patches", d.v.s. lapper, reparationer eller tilføjelser, (somme tider sikkerheds-rettelser) - eftersom denne http server (som så mange andre projekter) er udviklet ved at man har tilføjet den ene forandring efter den anden. Udviklerne sender *patches* til Apache Foundation, som godkender og indlemmer i projektet.

> *HTTPD er et af de mest succesrige Open Source projekter. Det er en server, som bruges på mange store sites med tunge webservices.*
>
> *Da IBM ledte efter et HTTP serverprodukt og erkendte at Apache var langt den bedste og på det tidspunkt næsten enerådene, var den første hurdle at finde ud af, hvem man skulle forhandle med. Det viste sig jo som bekendt at være en gruppe, som ikke var organiseret i et firma - og så fandt man ud af at man ikke kunne købe et Open Source projekt og besluttede i stedet at støtte det. Sidenhen har IBM branchet fra Apache og bygger en IBM HTTP server over Apache-koden (hvilket jo er helt legalt). IBM støtter også andre Open Source projekter, især nævnes Linux kernen og X-Windows.*
>
> *IBM sælger support til deres branch af Apache, men frigiver ikke den kode, de selv udvikler. IBMs tilføjelser*

omfatter bl.a. server side caching, d.v.s. at de websider (eller bare webside-objekter) som oftest bliver requested, bliver dannet én gang og genbrugt.

Det kan man også gøre med Apache og programmet Varnish, en server side cache. Varnish er udviklet af Poul Henning Kamp og er et projekt, som har vundet verdensomspændende anerkendelse. YR.no er et af mange sites, som bruger varnish.

Apache installation

Når man installerer Apache fra kildetekst, source, hedder programmet httpd, og program-kildetekst arkiv hedder httpd-2.4.7-tar.bz2, hvor 'd' står for demon, d.v.s. baggrundsproces, en proces, som ikke stikker hovedet frem på en skærm, men meddeler sig gennem logfiler.

Installerer man fra source, kan man lægge hele molevitten i et filtræ, typisk `/usr/local/apache2` eller `/apache,` som kan lægges på en hurtig mounted disk, så har man modulariseret sin installation på en meget nem og overskuelig måde.

Ja! jeg ved Debian pakkere kan blive lidt fortørnet over dette synspunkt. Debians pakker er meget stabile og meget nemme at have med at gøre, så det anbefales at bruge dem, med mindre man har særlige behov!

Hvis man installerer fra Debian-pakker, kommer programmet til at hedde apach2, og filerne spredes (som sædvanligt) i editerbar text configuration (/etc), library, binaries, dokumentation og webfiler (websider).

Fra source

Installation af en Apache httpd server kræver mere end én source pakke. Jeg hentede til denne artikel:

httpd-2.4.7.tar.bz2

apr-1.5.0.tar.bz2

apr-util-1.5.3.tar.bz2

Kompilering består af flg. operationer - vi antager at source-pakker er til rådighed i **`/distfiles`**:

```
cd /usr/local/src

tar -xkjf /distfiles/httpd-2.4.7*
cd httpd-2.4.7
tar -xkjf /distfiles/apr-1.5.0*
tar -xkjf /distfiles/apr-util-1.5.3*

mv apr-1.5.0 srclib/apr
mv apr-util-1.5.3 srclib/apr-util

./configure --with-included-apr &&
make && make install
```

hvor sidste led (make install) er mere end kopiering af httpd binary til et program-directory. Antallet af installerede filer løber op til 1,781 som fylder 30 MB.

Hele installationen ligger efter denne source-installation i **`/usr/local/apache2`**

Installation af Apache2 på Debian og derivater (Ubuntu, Mint)

Der er binary-pakker med Apache webserveren og moduler på de fleste Linux-distroer.

Hvis man bruger Debian (den mest udbredte og stabile server-distribution) skal man lige være klar over at Apache httpd hedder apache2 og at Debian har opdelt den i mange mindre pakker, ligesom Debian gør med andre projekter.

Følgende anvisning er kun vejledende. Man kan formentlig komme ud for at éns Debian installation mangler flere andre pakker end de her anførte, og nogle systemadministratorer vil sikkert mene at man skal gøre tingene på en anden måde. Følgende kommandolinie trækker det meste ind på min atypiske installation:

```
# apt-get install libapache2-mod-php5
[...]
The following NEW packages will be
installed:
  apache2-mpm-prefork apache2-utils
apache2.2-bin apache2.2-common libapache2-
mod-php5 libapr1 libaprutil1 libaprutil1-
dbd-sqlite3 libaprutil1-ldap php5-cli
php5-common

# apt-get install php5-pgsql
```

Navne baseret virtuel server

Hvis vi kører på et typisk hjemme net med en router fra en ISP, som fx. Telenor eller TDC, har éns computer ikke et synligt IP-nummer, og sikkert heller ikke DNS, så da kan man springe næste afsnit over.

Hvilket IP-nummer har udviklingsserveren på hejnettet?

Kører vi på et globalt synligt IP-nummer, og vil vi være sikre på, at denne nameserver ikke ses udefra, bør man selvfølgelig spærre for adgang til port 80. Men der er faktisk en anden måde man kan lave en test-server - og samtidig have et synligt website kørende. Man kan lave en navnebaseret virtuel server, som fx. kaldes ***udviklingsserver.hejnet***. For at éns browser kontakter denne "server" skal man i sin /etc/hosts fil indskrive IPnummeret (som gerne må være lokalt, privat) og give det navnet "udviklingsserver.hejnet":

```
192.168.1.10      <rigtige-servernavn>
192.168.1.10      udviklingsserver.hejnet
```

Når man indtaster "udviklingsserver.hejnet" i location linien (URL-feltet) vil browseren først kalde ***gethostbyname(3)*** som "ekspederes" af libc (systemets centrale library).

Afhængigt af `/etc/dnsswitch.conf` får man svar fra den ene eller anden kilde først, normalt først fra /etc/hosts. Derfor vil browseren få adresse 192..168.1.10 som svar på vores spørgsmål *"hvilket IP-nummer har udviklingsserveren på hejnettet?"*.

Når man er klar med sin virtuelle server, må man oprette en DNS record, så hele verden kan se, hvilket IP nummer den har. Derefter begynder der at komme trafik til serveren.

Hvorfor det virker

HTTP, Hyper Text Transfer Protocol har til forskel fra andre Internet protokoller navnet fra location-linien med i den første datapakke, som sendes.

Ud fra navnet i denne pakke kan serveren se, hvilken virtuel server, som browseren (brugeren) ønsker skal svare.

På den måde kan Apache svare forskellige ting på forskellige domæne, selv om den kun har ét IP-nummer. Fx har **`fmkj.dk`** og **`webhotel2.ruc.dk`** samme IP-adresse - men svaret fra serveren er forskelligt.

I Debian installationer skrives virtuelle hosts ind i separate filer: læg en fil i /etc/apache2/sites-available og lav symlink til /etc/apache2/sites-enabled. Hvis det kun er midlertidigt, kan man rette i /etc/apache2/apache2.conf nederst (men den kan blive overskrevet af pakkesystemet, hvis man opdaterer).

For at oprette en virtuel server skrives i httpd.conf en <VirtualHost> blok. Dokumentationen på httpd.apache.org er ganske udmærket, men på næste side er et eksempel for de utålmodige.

```
httpd.conf
<VirtualHost 192.168.224.144:80>
    ServerAdmin donald_j_axel@yahoo.com
    DocumentRoot "/usr/local/apache2/udv/htdocs"
    ServerName udviklingsserver.saxen
    ServerAlias udv

    <Directory /usr/local/apache2/udv/htdocs>
        AllowOverride none
        Require all granted
    </Directory>

    ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin
    <Directory "/usr/local/apache2/udv/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    <FilesMatch \.php$>
        SetHandler application/x-httpd-php
    </FilesMatch>

    ErrorLog /usr/local/apache2/logs/udv-error.log

    # Mulige: debug, info, notice, warn, error, crit,
    # alert, emerg. # Debian/Ubuntu defaults to warn.
    LogLevel info

    CustomLog /usr/local/apache2/logs/udv.log combined

</VirtualHost>
                                    1,0-1         Top
```

Som det ses er der oprettet et særligt directory for den virtuelle server, /usr/local/apache2/udv og her lægges de filer, som serveren skal "servere" for gæsterne.

Apache konfigurationen består af direktiver - et direktiv er en kommando bestående af type og værdi, som fx.

VirtualHost ***IP-nummer:port***

Man kan skrive nøgleordene (type-angivelse) med store eller små bogstaver. Det er med andre ord ikke en fejl at skrive "vitualhost" med lutter små bogstaver.

VirtualHost danner en blok, og inden i den sættes så værdierne, der skal gælde for denne virtuelle server.

Vi har brug for at angive serverens navn. På httpd.apache.org er der et særligt afsnit med råd angående navne-baserede virtuelle servere, som denne, der ikke har sit eget IP-nummer.

En virtuel server har brug for at vide, hvad den hedder - selvfølgelig! - og hvor dens filer er, desuden regler for, om den må køre Common Gateway Interface (CGI) programmer, som jo straks åbner for en række sikkerhedsproblemer. Det er imidlertid netop CGI, som gør det muligt at lave dynamiske websider.

Desuden angives en særlig logfil for denne server. Det bruges naturligvis på webhoteller, hvor den enkelte bruger ikke skal kunne se alle de andres logging.

Dynamiske websider

CGI programmer kan være skrevet i hvilket som helst programmeringssprog, det kan være binaries og det kan være fortolkede programmer. Det mest almindelige er fortolkede PERL programmer, eller Python programmer. PHP er lidt anderledes, fordi PHP kode kan ligge indlejret (embedded) i HTML.

PHP programmer kan også godt ligge som CGI-filer, men det er mere almindeligt at bruge PHP som blokke inden i HTML sider. I konfigurationen af vores lille virtuelle server skriver vi derfor en blok som angiver, at *.php - filer skal sendes igennem php-modulet, der udfører indlejrede programblokke.

Resultatet, output i form af text fra program-blokkene, lægges ind i HTML dokumentet der, hvor program-blokken lå.

Et minimalt PHP program til Apache-2.4.7 og PHP-5.5.8 ser således ud:

```
<html><head><title>Mini-PHP</title></head>
<body>
<?php
        phpinfo();
?>
</body></html>
```

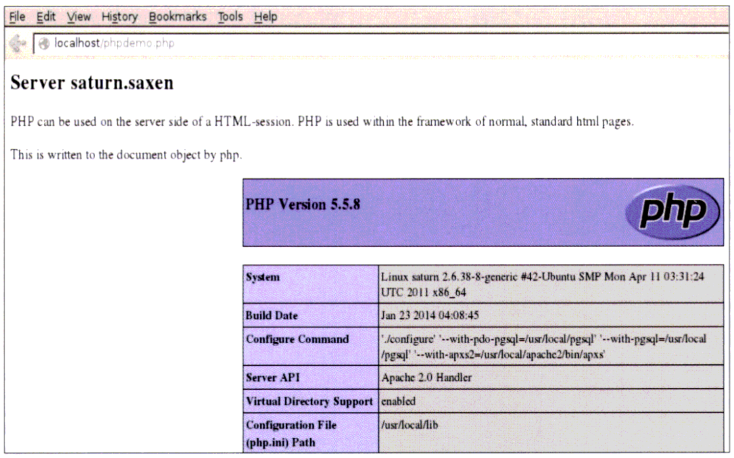
Inde i HTML-body blokken er der ikke andet end en markør for PHP-kode start, **<?php** hvorefter en funktion kaldes. Funktion hedder phpinfo, og den returnerer en tabel i html-format, som indeholder en lang række oplysninger om hvad PHP kan. Det er en meget nyttig funktion, hvis man fx. vil vide om PostgreSQL database-funktioner forefindes og er klar til brug.

Der skal semikolon efter hver sætning (statement) og hvis det mangler, konstaterer PHP-modulet at der er syntax fejl og kører ikke koden.

Slutmarkøren er **?>**

Da fortolkning kræver en del CPU, har et stort site som Facebook brug for en variant af PHP, som kan kompileres og køre som binaries. Det fandtes ikke, så det måtte de selv lave.

Men for et mindre site med 10 hit i sekundet vil man kunne få tilfredsstillende hastighed med en opsætning, som bruger fortolket kode.



phpinfo() afleverer meget nyttig information, meget mere end vist i illustrationen ovenfor (som er et crop af et screen-dump) blandt andet kan man se PHP-version og konfiguration. Derfor afslører screen-dumpet, at det er en PHP på min server, som er configureret med pgsql (postgres) men ikke med MySQL (eller MariaDB, NoSQL etc.) og at den har forbindelse til Apaches eXtenSion tool interface, apxs; det er den mekanisme, som gør det muligt at bygge (kompilere) og installere moduler, som kan loades i Apache httpd med kommandoen "loadmodule".

Det er med i httpd som standard, også når man installerer fra source med ***apr***, Apache Portable Runtime.

Kontroller med en "detektivkommando":

```
sat2:/arb173/#httpd -l
Compiled in modules:
  core.c
  mod_so.c
  http_core.c
  event.c
sat2:/arb173/#
```

<>

Forbindelse til databasen

Nu er vi klar til at programmere web-applikationer. Næsten! Vi skal selvfølgelig også have en database, og vi skal sikre os at vi kan nå databasen fra php-applikationerne, og vi skal iøvrigt også gøre installationen mere hacker-sikker - det bliver emne for en senere artikel.

```
pgdata=# create table dofobs (fugleart integer,
antal integer);
CREATE TABLE
pgdata=# \d
        List of relations
 Schema |  Name  | Type  | Owner
--------+--------+-------+--------
 public | dofobs | table | pgdata
 public | users  | table | pgdata
(2 rows)

pgdata=# insert into dofobs (17,1);
ERROR:  syntax error at or near "17"
LINE 1: insert into dofobs (17,1);
                           ^
pgdata=# insert into dofobs values (17,1);
INSERT 0 1
pgdata=# select * from dofobs;
 fugleart | antal
----------+-------
       17 |     1
(1 row)

pgdata=#
```

Psql er meget hjælpsom, hvis man ikke lige kan huske syntax. I eksemplet skal man notere sig, at der enten er en fejl, eller at der mangler et keyword, som det ofte sker for mig.

Som beskrevet i tegningen på side 8 er det PHP, som opretter en forbindelse til databasen.

Til det formål bruges **`pg_connect()`**. Forbindelsen sørger for, at vi kan sende en tekst-streng til databasen,

Kunne man oprepå andre måder? Ja, sagtens. Et shell script kunne være CGI program og udføre en SQL kommando via databasens klient-program. (Det gælder både Postgres, MySQL og så videre).

Fordelen ved at bruge embedded PHP kode er, at det fungerer fint sammen med HTML formatering.

pg_connect()

```
<?php
$conn=
pg_Connect("host=jupiter
hostaddr=192.168.224.143 dbname=pgdata
user=pgdata password='secret'");
?>
```

Her ses flere vigtige syntax-regler:

En PHP kodeblok indledes med en HTML-agtig tag, med spørgsmålstegn og sprog-navnet php.

En variabel har dollartegn som første character - ligesom i Perl og visse basic-dialekter.

Funktionsnavne er case-insensitive. Det er et levn fra de tidlige versioner - deprecieret, man skal ikke gøre det mere, helst ikke.

Funktions-parameter kan være en konstant-streng.

En streng/string kan angives med dobbelt-quote, og inden i den kan man bruge single-quotes. Det omvendte er også tilladt, ligesom i Javascript.

En string med space i **skal** quotes, men ellers er string-værdier inde i parameterstrengen ikke nødvendige at quote.

En sætning (kald til funktion) skal slutte med semikolon. En parameter string til en funktion må gerne deles over flere linier. Det er godt for læseligheden.

Endvidere ses det, at vi kan connecte via TCP/IP ved at angive host-navn og IP-nummer (parameter hostaddr).

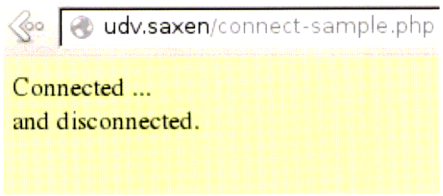
Så let er det faktisk at have databasen på en anden maskine på det interne net.

Hvis vi vil vide, om det lykkes at connecte til databasen, **skal** vi bruge den værdi, som funktionen returnerer.

Her er et lidt mere realistisk eksempel:

```
<html><head><title>Sample-connect</title></head>
<body>
<?php
    $conn = pg_connect("host=jupiter
    hostaddr=192.168.224.143
    dbname=pgdata
    user=pgdata
    password='my secret'");
    if (! $conn) {
       echo "Not connected";
       exit(123);
    }
    echo "Connected ...";
    sleep(1);
    pg_close($conn);
    echo "<br/> and disconnected.";
?>
</body></html>
```

Output fra ovenstående program vil (i bedste fald) være sådan:

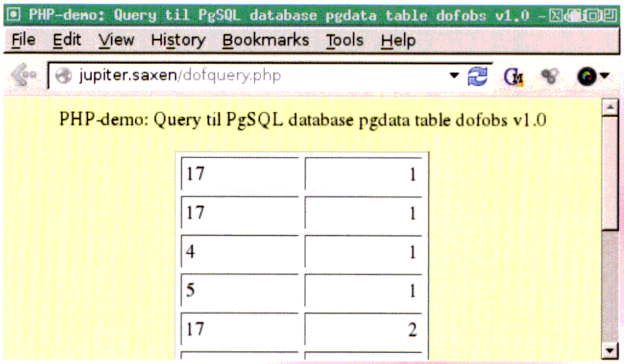


sleep-funktionen er selvfølgelig ikke nødvendig, men blot indlagt for psykologiens skyld - og for at demonstrere et andet, simpelt funktionskald.

Når vi har fat i databasen, kan vi liste indholdet af en tabel:

```
<html><head><meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title> Query dofobs v1.0</title></head><body>
<?php
  $conn = pg_connect("host=jupiter
hostaddr=192.168.224.143 dbname=pgdata
user=pgdata password='uha'");

  if (!$conn) {
     echo "An error occurred.\n";
     exit;
  }

  $result = pg_exec($conn,
     "SELECT * FROM dofobs;");
  if (!$result) {
     echo "An error occurred.\n";
     exit;
  }

  $num = pg_NumRows($result);
  $i = 0;

  echo "<TABLE BGCOLOR=#efefef BORDER
CELLPADDING=3 CELLSPACING=4 COL=3
ALIGN=CENTER>";
  while ($i < $num) {
     echo "<TR><TD width=90>";
     echo pg_result($result, $i, "fugleart");
     echo "</TD><TD width=90 ALIGN=RIGHT>";
     echo pg_result($result, $i, "antal");
     echo "</TD></TR>";
     $i++;
  }
  echo "</TABLE>";
  pg_freeResult($result);
  pg_close($conn);
?>
</body></html>
```

Output af denne simple query er formateret med table options - bare for at vise, at man ***kan*** formatere indholdet. Med cascading style-sheets (CSS) kan man opnå skønnere web-rendering end dette:



Som øvelse bør man tilføje kolonne-overskrifter og udvide databasen med et felt til tidsangivelse (timestamp) og opslag i arts-register, men i artikel-sammenhæng er det bedst at holde kode nede på et absolut minimum.

Nu må vi gennemgå det absolut vigtigste element i listningen, nemlig hvordan data kommer fra databasen over i variable, som er til rådighed for PHP:

```
$result = pg_exec($conn,
"SELECT * FROM dofobs;");
```

$result er et array af rækker (rows, databse-records). Alle array i php er associative arrays ligesom i andre fortolkede sprog (awk, javascript).

Her afsløres det, at denne artikels forfatter har haft fat i mange tidligere versioner af PHP: pg_exec() hedder nu pg_query() - men interessant nok fungerer det stadig at man kalder den pg_exec().

Her skal nævnes at det er tilrådeligt at angive MAX antal rækker, som ønskes indlæst, ellers kan query'en tage lang tid og give for stor belastning på en "arbejdsom" server, fx:

```
select * from dofobs LIMIT 20;
```

Når man så vil have fat i værdierne i $result arrayet, løber man det igennem, fx.:

```
echo pg_result($result, $i, "fugleart");
```

hvor $i er et index og "fugleart" skal være felt-navnet fra databasen. D.v.s. at al information om databasens schema er hardcode i de her givne eksempler.

For at index (tælleren) skal holde sig inden for antallet af fundne records, skal vi først have fat i antal:

```
  $num = pg_NumRows($result);
```

Funktionsnavne er som nævnt case-insensitive, og jeg foretrækker "kamelstavning" fordi det øger læseligheden.

En løkke som i C, Perl eller Javascript:

```
$i=0; while ($i < $num){
   echo pg_fetch_row($result,$i,"fugleart");
}
```

Resten er pynt, ren layout så listningen ser pæn ud - dog er

```
  pg_freeResult($result);
  pg_close($conn);
```

ikke helt overflødige, fordi de selvfølgelig frigør ressourcer.

Indtastning af data

Her er kode eller HTML til et minimalistisk input skema. HTML forms er ikke meget værd uden Javascript eller ***client side scripting***. Det ses af en af de valgmuligheder, options, som angives i **form-taggen: ACTION.**

Data fra indtastningen lejres som en variabel med navn som angivet i inputfeltet. Denne variabel, eller rettere navn+indhold, sendes via en text-string til det program, som angives med sti eller path. For at kunne det rigtigt, skal man have overblik over, hvordan apache håndterer directory-navne.

```
<form method="post" action="/path-til-progam">
<input name="variabelnavn">
</form>
```

I dette eksempel er */path-til-program* med skråstreg (slash) foran angivelse af at apache serveren skal finde et program i den virtuelle servers document-root. Med andre ord, full-path begynder ikke fra operativ-systemets filtræ-rod, men fra den virtuelle servers document root, som angivet i konfiguration af webserveren.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD>
  <META CONTENT="http-equiv="
  Content-Type" content="text/html; charset=ISO-8859-15">
  <TITLE>Tilføj Database Record</TITLE>
</HEAD>

<BODY>
<DIV ALIGN="center">
<H1>Add Database Record</H1>
<!-- FORM: "action" kontrollerer hvilket
program, der bliver udført, når brugeren
trykker på "send"knap eller trykker enter
i sidste felt./-->
  <FORM METHOD="post" ACTION="/dofinsert.php">
      <TABLE BORDER="0">
        <TR>
          <TD>Art</TD>
          <TD><INPUT NAME="i_art"></TD>
        </TR>

        <TR>
          <TD>Antal:</TD>
          <TD><INPUT NAME="i_antal"></TD>
        </TR>
      </TABLE><INPUT TYPE="submit">
    </FORM>
  </DIV>
</BODY>
</HTML>
```

Metoden, METHOD="post", angiver hvordan data skal oversendes. Her bruges post, som sender datastrømmen som input til programmet. Der findes en anden metode, "get", som danner en lang URL (eller URI) hvor data følger efter selve server-adresse+filnavn, fx.

http://dk1.php.net/manual-lookup.php?pattern=pg_select

Det er de lange URL'er som man ser, når man fx. bladrer i sin browsers history der viser, hvilke websites (filer) man har hentet. Hvis man prøver samme URL med en lille ændring:

http://dk1.php.net/manual-lookup.php?pattern=pg_result

opdager man at php-manualen ikke kender funktionen pg_result, og at søgefunktionen (manual-lookup) derfor viser en webside med oversigt over de nuværende, anbefalede funktionsnavne. (Det vil ved nærmere granskning vise sig, at den tidligere benyttede funktions-identifier *pg_result* er omdøbt til *pg_fetch_result*.)

Med andre ord skal vi nu til at skrive et program, som læser input fra form og sender det videre til databasen, der skal oprette en ny record (en ny row) i databasen.

Overførsel fra input til PHP variable

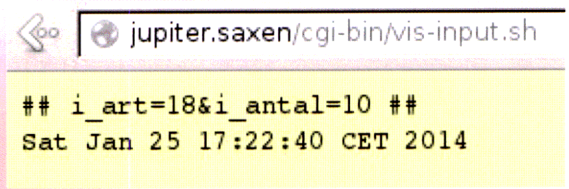
Man kan, hvis man vil se det (og det er ganske sundt) lave et lillebitte shell-program, som viser hvilke data browseren sender tilbage til apache serverprogrammet og hvordan apache sender det videre til vores lille program.

I så fald skal man lave et CGI program, og ***action*** skal ændres fra *dofinsert.php* til fx. ***/cgi-bin/show-input***. Programmet kan være skrevet i hvadsomhelst, der kan læse fra standard input - fx. C eller C++ - eller det kan være et simpelt shell-script, som læser fra stdin med kommandoen ***read:***

```
#!/bin/sh
echo "Content-type: text/plain\n";
read INPLIN
echo "##" $INPLIN "##";
date
```

Hvis man er lidt nysgerrig efter at se det fungere, skriver man også nogle non-destruktive kommandoer ind i dette minimale CGI-program, i eksemplet en date-kommando.

Output kan fx. se sådan ud:



Hvis man nu skrev en kommando, som gjorde noget destruktivt, ville den så blive udført? fx. **`rm -f *`** ? eller noget værre? - **Ja,** det er en af grundene til, at en webserver ikke skal køre som superuser og at den iøvrigt skal sikres omhyggeligt.

Bemærk at data kommer som felt-navn, lighedstegn, værdi, hvorefter der følger et "og-tegn" (ampersand), næste navn/værdi-par og så fremdeles. Det format kaldes **application/x-www-form-urlencoded** og beskrives i RFC1738 sektion 2.2.

Det er den måde, data kodes på hvis der ikke angives andet (default encoding). Reglerne er, at kun US-ASCII må bruges, svarende til bogstav-koder fra (decimal) 32 til 126 og alt andet erstattet af %HH, procent-tegn og to hexadecimale cifre; derfor kaldes det populært *percent-encoding.* Fx. kodes CR LF (Carriage-Return LineFeed, linieskift) som %0D%0A.

Afkodning af dette skal foretages af PHP-script. PHP danner automatisk et associativt array kaldet _POST, hvor værdien af _POST["i_art"] og antal _POST["i_antal"].

(Identifierne har i_ forrest (*i_art* og *i_antal*) for at minde om at de kommer fra input-felterne i HTML-formen.)

I det minimalistiske eksempel er der eksempel på print debugging: Allerførst vil vi gerne se, at programmet kører.

```
echo "dofinsert is running";
```

Jeg bruger normalt engelske kommentarer mv. fordi vi engang oplevede at et større system ikke kunne exporteres til Letland fordi det var på dansk - men nogle af de engelske brokker har jeg dog fået luget ud i dagens anledning.

Næste linier bruger *var_dump()* til at vise, om vi har fået variablene med over (og indholdet af dem). Hvis det nu skulle være et drifts-program skulle der naturligvis være range-checking - gerne i en blok som fremtræder klart læselig i koden.

Men vi nøjes med at printe indholdet nu. Da *var_dump()* ikke laver html kode, indsætter vi en HTML-tag: **`<pre>`** som betyder *preformateret text,* som browseren skal vise som plain text, d.v.s. den skal bevare linieskift.

Resultatet ses i illustrationen under php-programmet: skriftsnit (font) skifter til monospace (courier). _POST er et array(2) i dette tilfælde, hvorefter hvert element vises.

```
<HTML><HEAD><BODY>
<?php
  echo "dofinsert is running.\n";
  echo "<pre>";
  var_dump($_POST);
  echo "</pre>";
  $fugleart = $_POST["i_art"];
  $antal = $_POST["i_antal"];
  echo "Fugleart: ";
  echo $fugleart . "<p/>";
  echo "Antal: ";
  echo $antal . "<p/>";

  $conn = pg_Connect("host=jupiter
    hostaddr=192.168.224.143
    dbname=pgdata user=pgdata
    password='my secret'");
  if (! $conn) {
    echo "Database connection refused.\n";
    sleep(6);
    exit;
  }
  $success = pg_exec($conn,
    "INSERT INTO dofobs (fugleart, antal) VALUES ('$fugleart', $antal);");
  if ($success) {
    echo "<p>Observationer opdateret! ";
    echo "Exit-kode: $success.\n";
  }
?>

<p /><a href="http://jupiter.saxen/dofquery.php">Print liste ...</a>
<p /><a href="/dofinput.php">Tilbage ...</a>
</BODY>
</HTML>
                                                    34,1          Top
```

For læselighedens skyld (og for øvelsens skyld) hives indholdet af de to array-elementer nu over i to almindelige variable, bemærk dollar-tegnene før variablene, både _POST og de lokale variable som har scope i hele program-forløbet.

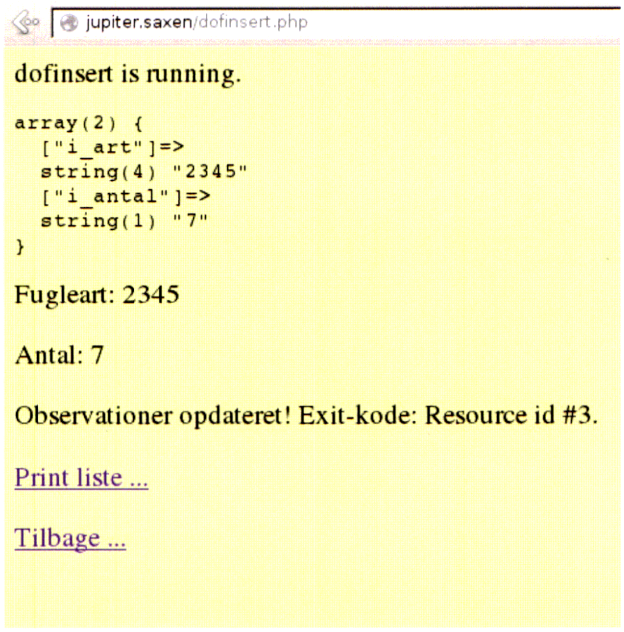
Så følger - endelig - oprettelse af forbindelse til databasen, der som sagt ligger på en anden server, derfor er hostnavn og hostadresse nødvendige. Connection string er en række nøgleord og værdier, og det fremgår at man gerne må dele en string over flere linier (modsat C-syntax).

Den efterfølgende pg_exec string er ren SQL med den undtagelse at lokale variable gerne må bruges inde i denne string. Det ses let at det er uhyre praktisk!

Der testes for, om insertion er gået godt, og det meddeles - igen print-debugging, som selvfølgelig fjernes, når programmet er færdigt og skal sættes i drift.

De to sidste linier kan erstattes, så programmet ikke venter på yderligere input, men fortsætter med en anden webside.

Her ses resultatet:



■

DKUUGs foredrags video system

Af: Jon Bendtsen, jon.bendtsen@jonix.dk

Denne artikel vil beskrive det video system i DKUUG som jeg nåede at opsætte, konfigurere og bruge et par gange før jeg sammen med Bryan Østergaard og Alexander Hansen Færøy forlod DKUUG og Open Source Days ApS i foråret 2013.

Det er en længere artikel inddelt i flere afsnit. I 1. afsnit vil jeg starte med at definere formålet og problemstillingen. I 2. afsnit vil jeg fortælle om mine og andres erfaringer med video optagelse, efterbehandling og udgivelse. I 3. afsnit vil jeg fortælle om hvilken hardware jeg overvejede, men forkastede, samt hvad jeg endte med at foreslå at DKUUG indkøbte. I 4. afsnit vil jeg beskrive proceduren ved optagelse. I 5. afsnit vil jeg beskrive efterbehandlingen. I 6. afsnit vil jeg runde af med at beskrive publiceringen.

Formål og problemstilling:

Formål: publicere video på nettet af (DKUUG) foredrag med god lyd og billede

Problemstillinger:

Ofte kun en taler, men der kan være flere

Taler kan skifte imellem stående og siddende position

Ofte viser taler ikke kun en præsentation, men også kommando prompter og configurations filer

Ofte peger taler på det projekterede skærm billede med hans arme og hænder eller en laser pointer

Ofte mange tilhørere der alle kan stille spørgsmål, kommenterer

Ofte sidder tilhørerne ned, men er der mange så er der nogen som står op

Lokalet kan være ret fyldt

Ofte kun 1 bemanding m/k der skal klare det hele

Ofte kommer tilhørerne dryssende

Ofte udluftning og anden støj i bygningen under foredraget

Sjældent en dedikeret person m/k til at betjene video/lyd under optagelse

Ofte optages der om aftenen, så ingen bliver og editerer videoen

Redigering af videoen tager mindst lige så lang tid som optagelsen af videoen

Foredraget kan blive afbrudt af 1 eller flere pauser

Fjernelse af støj i efterbehandlingen er i praksis helt umulig

Et foredrag kan godt tage op til 3 timer

Erfaringer

I gennem en år række var der flere forskellige frivillige i DKUUG der arbejdede på at optage, efterbehandle og udgive video fra DKUUGs foredrag. Når det lykkedes for mig at skabe et godt system så skyldes det 5 ting:

1. Jeg kendte de andres erfaringer
2. Jeg blev ved med at arbejde på opgaven indtil den blev løst
3. Jeg faldt tilfældigvis over den nøgle komponent der gjorde fjernredigering meget lettere.
4. Jeg arbejdede og afprøvede alle dele af hele processen fra optagelse hen over efterbehandling til udgivelse, de andre prøvede kun dele af opgaven
5. Jeg har direkte adgang til et professionelt video team der igennem mange år har optaget, redigeret og udgivet videoer professionelt

For neden har jeg listet de erfaringer som andre frivillige og jeg har gjort mig omkring optagelse af video på foredrag.

- Zoom er nødvendigt
- Stativ er nødvendigt
- Stort set alle (HD) kameraer giver et godt (nok) billede
- god lyd er svært, lav tale, høj støj, hosten, brummen og summen
- talermicrofon er nødvendig
- microfon til spørgsmål/kommentarer fra salen er en god ting at have, men tager lang tid at flytte rundt, og kræver en eller flere opmærksomme personer til at række microfonen rundt for tilhørerne gider ikke vente på microfonen før de stiller deres spørgsmål
- Tilhørerne sidder ofte så tæt at du ikke får dem til at gå hen til en dedikeret spørge microfon
- Taler glemmer at gentage spørgsmålet
- Talers lyd er langt den vigtigste da det er 99% af tiden
- for mange microfoner giver for meget extra efterbehandlingsarbejde
- video kamera skal optage taler + projektor billede ved siden af, da taler peger, artikulerer, ...
- autogain på lyden er skadeligt fordi selv i de få milisekunder imellem talers ord, så skruer kameraet op for gain på lyden, og derved vil taler mikrofonen også optage resten af lyden fra rummet.
- consumer kameraer kan ikke slå autogain fra, nogle prosumer kan
- optagelse af lyden på særskilt device giver syncing issues + mere besvær
- man glemmer at klapper foran kameraet har man noget at klippe efter når lyden skal synkroniseres
- konstant optagelse i op til 4 timer
- klipning og efterbehandling tager tid, lang tid, helt utrolig lang tid
- download og specielt upload af HD på via en *DSL er urealistisk
- mange dårlige video behandlingssystemer
- vi skal egentlig kun klippe start, pauser og slut
- DSLR optager under 30 mins video (pga. told på video udstyr)
- Professionelle kameraer optager MANGE GB data
- 220 volt kan blive afbrudt af en ledning der rives ud
- Batterier kan løbe tør
- let transport i få dele
- optagelse på udskiftelige hukommelses kort er at foretrække

- lyd og billede er ikke altid helt perfekt synkrone efter klipning og anden behandling, specielt ikke hvis man har optaget lyd og video hver for sig

Den afprøvede hardware, software og procedurer

Diverse gamle kameraer der optager på DV bånd, små flash kort, små harddiske, lav opløsning, ...

Canon consumer HD med Auto gain på lyden der ikke kan slås fra

Wireless microfon med en stor basestation der kræver 220 volt

Kun microfon på taler

Dårlige stativer, dårlig placering pga. rummets indretning

kino til redigering, cinelerra, ...

Lydoptagelse ved siden af på zoom h4n

Lejet udstyr. Ændret i sidste øjeblik. FYLDER meget. Udstyret gjorde ikke som forventet, men gav en god lyd og et godt billede

Minimums udstyrs krav:

- Kamera skal have zoom og kunne monteres på et stativ
- Kamera skal kunne fungere både på 220v og batteri på samme tid
- Kamera skal have 2 lyd kanaler, en til taler og en til rummet
- Kamera skal have aktive XLR stik der kan levere strøm til microfon
- Video skal ikke fylde alt for meget
- Kamera skal kunne optage i mindst 4 timer uden afbrydelse
- Udstyret skal være let og hurtigt at opsætte

Man får en kæmpe fordel ved at have talers lyd i det ene spor og lyden fra rummet i det andet spor, fordi hvis talers lyd går galt, så har man altid den fra rummet som backup, og hvis man har god taler lyd, så kan man let mute lyden fra rummet når der ikke stilles spørgsmålet eller er kommentarer. Derved minimerer man muligheden for støj på optagelsen.

Hardwaren

Den forkastede

DSLR kameraer blev forkastet fordi de ikke optager mere end 30 minutters video og ikke har XLR stik

Diverse Sony semi professionelt Video kameraer i NEX serien blev forkastet fordi de ikke havde XLR stik

JVC og Panasonic semi professionelt blev forkastet fordi de ikke kunne køre 220v og batteri på samme tid

Diverse externe Video recorder bokse, bla. Atomos og Sound Devices PIX blev forkastet fordi de optager alt for høj kvalitet og bare er extra besvær ved opsættelse

Blackmagic Cinema kamera blev forkastet fordi data fylder alt alt for meget, kan ikke optage i 4 timer uden afbrydelse og det mangler XLR stik

Diverse trådløse lydmodtagere som får strøm via kameraet fordi de krævede rigtig store TV system kameraer

Bluetooth wireless lyd fordi bluetooth sender på frekvenser som alle andre i rummet også bruger

Den købte hardware

- Canon XA 10
- SENNHEISER 112PG3 MICROPORT KIT
- SENNHEISER ME-4 CLIPS MIKROFON
- RØDE NT55 MIKROFON der sidder på kameraet til at optage spørgsmål og kommentarer fra salen
-

Canon XA 10 kan både have 220v og batteri tilsluttet samtidig, og ved afbrydelse af 220v skifter den uden ophold til batteri drift.

Canon XA 10 har 2 aktive XLR stik som endda hver for sig kan slås til og fra, således at der kun er strøm på den kablede mikrofon, hvorimod at den trådløse Senheiser med indbygget batteri ikke får tilsendt strøm.

Sennheiser ME-4 CLIPS er en retnings bestemt mikrofon lavet til at passe sammen med SENNHEISER 112PG3 MICROPORT KIT. Modtageren blev sat fast oven på Canon XA 10 i "skoen".

RØDE NT55 MIKROFON er en kablet XLR mikrofon der blev monteret i den indbyggede mikrofon holder på Canon XA 10.

Det hele blev pakket ned i en PELICASE 1550 så det blev beskyttet imod transport skader.

Proceduren før og ved optagelse

Før optagelse er det nødvendigt at verificere at alt udstyret, kamera, stativ, mikrofon og SD kort findes. SD kortet skal endda have nok ledig plads.

1. Hent kamera og stativ i kælderen og transporter det til foredragssalen
2. Fold stativet ud og sæt kameraet oven på. Tilslut strøm til kameraet, evt. via en 220v forlænger kabel
3. Tænd kameraet og tryk på optag. Ja, begynd at optage allerede nu. Sørg for at SD kortet har tilstrækkeligt med plads til hele optagelsen
4. Indstil zoom så man kan se hele projektor billedet og taler
5. Verificer at batteri niveau i trådløs taler lyd er tilstrækkeligt
6. Lav lydcheck på de forskellige mikrofoner, både den kablede til rum lyd, og den trådløse til taler via høretelefonerne som tilsluttes til kameraet
7. Fortæl taler hvor han skal stå/sidde, put mikrofonen på taler og instruer taler i hvordan taler tænder og slukker for mikrofonen
8. Overvåg kamera og verificer at det virker lige som forventet lige efter starten af foredraget.
9. Overvåg kamera og lyd i løbet af foredraget

Nedtagelse er bare at pakke tingene væk på den dertil indrettede plads. Husk dog at tømme SD kortet. Det gode ved et SD kort er at det bare kan puttes ind i en vilkårlig SD kort læser og begynde upload til serveren imens man pakker resten ned. Derved er indholdet af SD kortet ofte uploadet når eller kort tid efter at man har pakket resten ned og sagt farvel til taler og tilhørerne.

Efterbehandling

Grundlæggende er efterbehandlingen af videooptagelser fra DKUUGs foredrag super simple. Der er kun 2 nødvendige trin der skal gøres, men det er også rart med det 3. skridt, intro og outtro på videoerne.

1. Klip uønskede dele af videooptagelserne fra, dvs. fra starten af optagelsen til starten af foredraget, evt. pauser skal klippes ud, og til sidst så stopper foredraget før videooptagelserne stopper
2. Mix lyden fra 2 spor til 1 mono spor. Her lægger sporet med talers lyd grundlaget. Men lyden fra rummet skal kun bruges når der er spørgsmål eller kommentarer fra salen. Derfor kan man i langt det meste af tiden mute lyden fra rummet, og de dele der ikke er mute kan man merge med taler lyden.
3. Det er rart med intro og outtro på videoerne der fx. fortæller hvad videoen handler om, dvs. titel + evt. en under titel; Hvem taler er; Hvor lang tid videoen varer; Hvilken licens den er udgivet under; Hvem der har lavet diverse frivilligt arbejde. Men det er ikke noget hårdt krav, for man kan jo navngive filen så den indeholder de oplysninger.

De rå optagelser

Canon XA 10 er et AVCHD kamera, dvs. det optager i mpeg 4 video codec. Container formatet er MTS, hvilket ikke er så godt beskrevet. MTS ligger i 2GB filer, så når den 1. fil når 2GB så forsættes der i en ny 2GB fil.

Da det er besværligere at behandle mange filer, så oversætter DKUUGs video system alle disse filer til en stor fil i høj kvalitet. Desuden så giver dette mindre risiko for synkroniserings issues imellem lyd og billede.

Jeg kalder denne store fil for kildefilen.

Pga. senere værktøjer så er denne store fil en .webm fil. Webm er et frit åbent format http://en.wikipedia.org/wiki/Webm, dvs. at jeg rent faktisk konverede både lyd og billede fra mpeg 4 til webm.

Lyd og video behandling er paralleliserbart hver for sig

Da lydmixning og videoklipning kan ske hver for sig, så exporterer jeg lyden fra den store kildefil.

Lyden behandles derefter i et vilkårligt lyd program, fx Audacity som var det jeg altid brugte. Stort set det eneste jeg gjorde var at splitte stereo sporet i 2 mono spor, og derefter mute sporet med rum lyd i de lange segmenter hvor tilhørerne ikke stiller spørgsmål eller kommer med kommentarer. Der var ingen grund til at mute før foredraget starter, i evt. pauser eller efter foredraget er slut, fordi de dele bliver alligevel klippet væk senere.

Nogle meget få gange var det også nødvendigt at prøve at rense taler lyden, men da Sennheiser ME-4 CLIPS er en retnings bestemt mikrofon så optager den stort set kun lyden fra taler og ikke anden støj, og da Canon XA kan bruges med manuel gain på lyden, så er det ofte slet ikke nødvendigt at behandle lyden, men hvis jeg havde god tid, så kunne jeg godt lide at mute ikke relevante lyde fra taler, fx: Øh, hosten, rømmen, trække vejret kraftigt, sukke, tsk, ..., osv.

Tilsidst mergede jeg de 2 lydspor til 1 mono spor som jeg så exporterede i de bedste OGG indstillinger så lyden kunne udskiftes i kildefilen.

Resultatet kalder jeg for lydokkildefilen.

Video klipning

For at kunne udlicetere video klipningen til 3.personer, gerne frivillige, medlemmer af DKUUG, foredrags tilhørere eller bare andre interesserede så bruger jeg et webbaseret Open Source video redigeringsværktøj kaldet MovieMasher. Dette værktøj er desværre baseret på Flash, men det gør også at stort set alle med en internet forbindelse nemt kan deltage i klippe arbejdet, altså crowdsourcing.

http://www.moviemasher.com/

Det geniale ved MovieMasher er at man gemmer redigerings arbejdet i en XML fil på serveren, så kan man nemlig bare parse denne XML fil og klippe i lydokkildefilen.

Moviemasher

Det tog lang tid at betvinge MovieMasher til at give en acceptabel performance. Klipning direkte i .MTS filer virker ikke, og selvom at man klipper i mindre mpeg 4 filer så fungerede det aldrig ordentligt. Det var også helt umuligt at få loadet MovieMasher på clienten hvis man prøvede at klippe i .AVI eller MJPEG filer.

Til sidst prøvede jeg med flash video filer, og det fungerede nogenlunde. Jeg begyndte derfor at optimere på flashfilerne således at jeg beholdte lyden i så god kvalitet som muligt, men at jeg reducerede opløsningen fra HD til 1/8 på begge leder, og så var den der. God performance, og når man kun skulle finde foredrags start og slut, samt evt. pauser, så fungerede det fint.

Proceduren jeg brugte til at finde klippe steder var denne:

1. på med hovedtelefoner og tryk på play
2. lad videoen køre i baggrunden til at lydniveauet skifter, dvs. at taler begynder foredraget eller begynder efter en pause (lavere), at en pause begynder (højere) eller at foredraget er slut (også højere).
3. Når jeg hørte et skift i lydniveauet så spoler jeg lidt tilbage et par gange indtil jeg finder det rette klippe tidspunkt.
4. Når det rette klippe tidspunkt er fundet, så trykker jeg på klippe knappen i MovieMasher, og bagefter på save knappen. Så bliver en XML fil opdateret på serveren, som jeg senere kan parse og bruge til at klippe i lydokkildefilen.
5. Jeg forsætter med at lytte efter næste lydskift, eller videoen er slut.

Automatisk klipning via mkvmerge og XML filen

Efter at klipningen er overstået så brugte jeg shell scripts til at parse XML filen således at jeg kunne bruge mkvmerge til at klippe i lydokkildefilen. Resultatet fra dette er en høj opløsnings høj kvalitets video af foredraget. Jeg kalder den for HQ videofilen og den fylder vel op imod 10GB.

Denne HQ fil oversætter jeg nu til 2 reducerede udgaver. Den ene er stadig i HD opløsning, men reduceret til ca. 2Mbit så resultatet kun fylder et par GB. Denne fil kalder jeg for 2M videofilen

Den anden reducerer jeg til 1/2 opløsning på hver led og kun nogle hundrede Kbit, således at LQ videofilen kommer ned og fylder nogle hundrede MB.

Intro og Outtro

For at kunne automatisere genereringen af intro og outtro så bruger jeg en .fig template så jeg kan bruge sed til at lave search and replace ud fra en lille fil med oplysninger om video. Dvs. Titel, taler, licens, dato, sted, chairman, kamera, klipper, mixer, ... osv. Desuden så beregner jeg også længden af en af de færdig klippede filer, HQ, 2M og LQ der naturligvis alle har samme længde.

Ved at bruge en .fig template så var det let at udvide .fig templaten med Open Source Days logoet da jeg efterbehandlede de videoer jeg optog på Open Source Days 2013.

Bagefter oversætter jeg .fig filen med transfig til en .png som jeg så klipper til i både HD format og halvdelen på hver led, for intro og outtro bliver ikke så pæne hvis man lader ffmpeg om at reducere dem.

Disse 4 statiske billedfiler HD intro, HD outtro, LQ intro og LQ outtro konverterer jeg så til en lille kort WebM fil på nogle få sekunder.

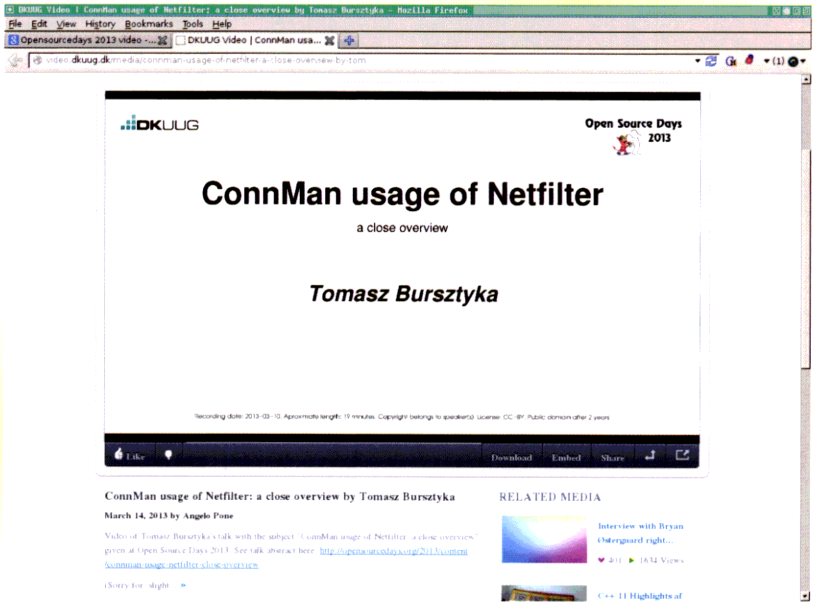
Igen bruger jeg mkvmerge til at samle:

HD intro + HQ videofilen + HD outtro = HQ_video_af_foredraget.webm

HD Intro + 2M videofilen + HD outtro = 2MHD_video_af_foredraget.webm

LQ intro + LQ videofilen + LQ outtro = LQ_video_af_foredraget.webm

og resultatet er 3 færdige videoer som er parat til udgivelse.



Udgivelsen

I bund og grund kan udgivelsen være super simpel. Det er helt tilstrækkeligt med en http eller ftp adgang til webm video filerne.

Mediacore Video Content Management system

Mediacore er et online webbaseret system der giver adgang til at se forskellige video filer via en online afspiller. Mediacore kan rate videoer, og hvis ikke det var fordi kommentar felterne bliver fyldt med spam, ja så ville det også være muligt at kommenter på videoerne. Mediacore er således en slags youtube for dine "egne" videoer. Mediacore kan afspille videoer fra http og ftp URLer, fra YouTube, fra Vimeo, …

DKUUGs mediacore installation kan ses på http://video.dkuug.dk/ hvor også video.opensourcedays.org peger hen.

Youtube

Da vi gerne vil have folk til at se vores foredrag, og der er en hel del mennesker på Youtube, så er det oplagt også at lægge videoerne der. Problemet er licens betingelserne. Youtube har 2:

- standard Youtube Licens
- CC BY attribution. Det er således ikke muligt at forbyde ændringer eller kommerciel brug

Archive.org

Foruden WayBack maskinen så har Archive.org faktisk også et voksende bibliotek af mange forskellige slags videoer, lyd og billeder. Og her på Archive.org kan man selv bestemme flere dele af sin CC licens, fx non kommercial og forbyde ændringer.

Konklusion

DKUUGs video system er endt med at opfylde formålet og løse alle problemstillingerne. Jeg håber derfor at nogen vil bruge systemet, enten i DKUUG eller også i andre situationer hvor man ofte gerne vil udgive videoer.

■

For 20 år siden

Klip fra DKuug-Nyt marts 1994

For 20 år siden var Linux og FreeBSD nyheder, gratis Unix til en PC - hardware for 10.000 kr. hedder det i redaktionens omtale af de to artikler. Poul Henning Kamp skrev om FreeBSD - hvilken hardware, man skulle vælge og hvis man ville have grafikken med, så kunne man komme op at køre for 20.000 kr. Man havde brug for 300 MB disk, hvis man skulle have det hele med. Det er ca. en milliontedel af den disk-kapacitet, der er på redaktørens private maskine i dag. Og den har kostet en tiendedel af det, vi måtte give for 20 år siden.

Bladene er scannet og ligger som PDF filer på www.dkuug.dk

Xpdf: /wb4/DKuug/dkuug-nyt-068.pdf

DKUUG-Ny

Nr. 68 — marts 1994

Gratis UNIX

Bare fordi noget er gratis behøver det absolut ikke være dårligt. DKUUG-Nyt undersøger i dette nummer om dette også gælder for UNIX-varianter. Vi ser på to forskellige:

Linux

René Seindal beretter om sine erfaringer fra 18 måneder med Linux

FreeBSD

Poul-Henning Kamp nærer stor modvilje mod UNIX-leverandører, bl.a. derfor er han med i FreeBSD-projektet

Page 1 of 28 125% Quit

Poul Henning Kamp skriver bl.a.:
*De fleste firmaer vil nu brokke sig over den manglende support, **men hvis man har en eMail forbindelse** og kan give en nogenlunde præcis fejlbeskrivelse, så varer det sjældent over et par dage før fejlen er rettet, hvis man da ikke selv griber den medfølgende source og fixer det.*

Redaktionens fremhævelse

DKUUG-Nyt nr. 68 15

FreeBSD 1.1

— Gratis UNIX til alle

Hvis man mangler, savner eller bare ikke har, en UNIX-maskine, kan man nu få en gratis UNIX til en PC. Det har man egentlig kunnet få i snart to år, men først nu er kvaliteten blevet således at man kan tillade sig at bruge den til andet end legetøj. I denne artikel vil jeg beskrive FreeBSD release 1.1, som skulle være klar omkring 20. PORT.TXT) Resultatet er en UNIX med fuld kildetekst, der kører på en PC'er.

Der er indgået en aftale med "Walnut Creek CD-ROM" således at de, mod at donere og forstå driften af en udviklings-maskine for projektet, har lov til at distribuere FreeBSD på CD-ROM. Planen er at der skal udkomme en ny version ca. fire gan-

hele tiden.

Som det ses, behøver det ikke koste en formue at lave en UNIX-maskine mere, en PC til en titusind kroner, så er man faktisk kørende. Man kan f.eks lave en glimrende og hurtig X-station med en 486 med VLbus, et godt grafikkort, og et ethernet kort, pris: under 20.000,- excl. monitor. Eller en lille NFS-diskserver: 35.000,- med 4 Gb disk.

Man skal altså til at tænke lidt anderledes nu, fordi man simpelthen kan stille en UNIX-maskine op hvis man har brug for det, var det måske ikke en god ide at få lavet en dedikeret print-server? en UUCP/mail-maskine? en Domain-name-server? eller slet og ret bare en sandkasse?

Kommandocentralen

Monitorering

Unix, Posix, Linux, BSD, NT, IOS osv: Alle IT-systemer har brug for monitorer, information om hvad der sker, hvilke programmer der kører og hvilke dele af hardwaren, der belastes. Det er stadig sådan, at disk-access koster tusinder gange mere tid end memory access, og man kan spare tid og penge ved at udnytte systemerne bedre.

Fra gammel tid har jeg kørt en xload for lige at holde øje med, hvor meget belastning - load - der er på CPU'en. Det mål, der anvendes, er ***run queue, kørsels-kø.***

Det beskrives i manual-siden for uptime programmet, at load er gennemsnit af antal processer, som kører eller som venter på IO ressourcer.

```
saturn:/ #uptime
 03:02:39 up 9 days, 20:16, 14 users,
load average: 0.22, 0.06, 0.06
saturn:/ #
```

Load average, belastnings-gennemsnit, angives for 1 minut, 5 minutter og 15 minutter.

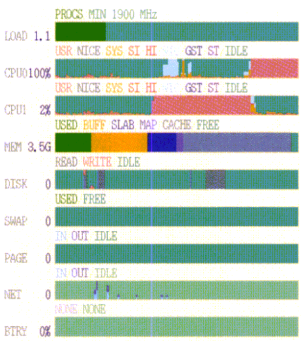
Men tallet tager ikke højde for, om maskinen har flere CPU'er eller en CPU med flere kerner, eller om der er variabel frekvens på CPU'en.

Det samme gælder vmstat som i de to første kolonner splitter op mellem runnable og IO-blocked.

Men som et kort overblik er det OK. Og ***Xload*** viser det så man med et blik kan danne sig indtryk af det.

xosview uden tilpasning

Sidder man på en ssh-forbindelse, kan man enable tunneling af X11, hvilket er den eneste rigtig sikre måde at køre remote host på Unix/Linux systemer. Den gamle ide om at Xload kunne bruge rwhod som informationskilde er tåbelig. Man kan på serveren enable X11 i /etc/sshd_config og på klienten i /etc/ssh/ssh_config. (Det samme gælder for alle X11pro-grammer, der ikke går direkte til skærmkortet, GLX).

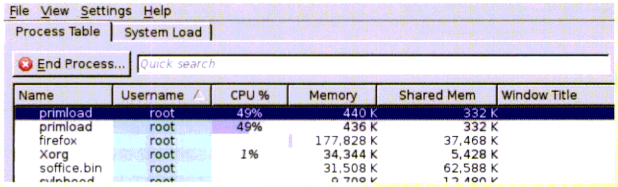


xosview m.ændrede farver

Men hvis man vil vide om det er net eller CPU eller disk, som giver problemer med systemets hastighed, må man selvfølgelig bruge andre programmer.

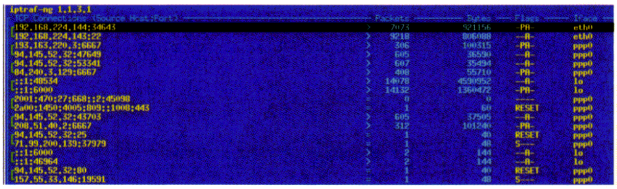
Mulighederne er mange, her omtaler vi kun redaktionens favoritter, xosview, ksysguard og net-overvågning.

Ksysguard er også en task- eller proces manager.



Den har mange lighedspunkter med Microsoft *taskmgr.*

Desuden er der panel-applikationer til windows-managere, både KDE, XFCE, Gnome etc. men skal man vide nøjagtig hvad det er for en trafik, som danner propper på nettet, må man bruge ***iptraf***.



Udsnit af iptraf statistik

Det er imidlertid bedre - omend større besvær - at installere en net-overvågning som fx. netsaint, der er kendt og afprøvet. Der er kommet alternativer, xymon, zabbix mv. som alle bruger en webserver som brugerinterface - så ved man den er tvær-platform og kan ses på alle devices, og om nødvendigt over det globale internet. Men de kræver selvfølgelig alle lidt ekstra indsats for at få dem op at køre.

CLMystery in Terminal City

Har man brug for at demonstrere for nevøerne at man kan lave mange ting med en kommandolinie - ting, som er umulige eller overmåde vanskelige med en GUI - så kan man tage et kig på ***clmystery.***

There's been a murder in Terminal City, and TCPD needs your help.

TCPD Terminal City Police Department.

Den er meget sød. - Hvis man lige fanger ideen først, begynder det at være en meget god parallel til at bruge kommandolinien for at søge efter fejl i logfiler mv.

Der er mange eksempler og minsandten om der ikke er både hints og en løsning, som er kodet i en form, der er morsom og elegant.

```
$ echo " " && sed -n '/Z/p' solution |
tail -n 6 | cut -c 8,22,23,41 && echo " "
J  B
E  O
R  W
E  E
M  R
Y  S
```

Men man skal ikke være forbavset, hvis barnet siger: Det er da meget nemmere at spille Grand Theft Auto San Andreas på min iPhone! ■

SUPERUSERS

FIND KURSUS

Vælg producent

Vælg teknologi

eller

Indtast søgeord

eller

2014

1984–2014

Bestil SuperUsers' nye kursuskatalog for 2014
på tlf. 48 28 07 06 eller mail super@superusers.dk