NTNU
Kunnskap for en bedre verden

# *Leksjon 7: Filbehandling*

*Svend Andreas Horgen, IDI Kalvskinnet, NTNU*
*Lærestoffet er utviklet for emnet IINI3003 Webprogrammering med PHP*

*Resymé: Vi begynner med en oppsummering av nyttige filkommandoer. Du bør ha lest bokas kapittel 8 (evt. skumme gjennom) før du setter i gang med denne leksjonen og tilhørende øving. Du vil få tips til hvordan du kan unngå feil i tellere, kutte ned på utviklingstiden knyttet til filbehandling, laste opp flere filer samtidig, og unngå at filer som lastes opp overskrives.*

# 1. Funksjoner for å jobbe mot tekstfiler

For å operere mot en fil må den åpnes i riktig modus.

- `r` lar deg lese fra filen
- `w` lar deg skrive til filen. Alt gammelt innhold slettes
- `a` lar deg legge til innhold på slutten av filen.

I tillegg kan en bruke en `+` bak bokstaven, og det indikerer både lesing og skriving. Det fins også et x-modus som ikke er særlig aktuelt å bruke.

Kommandoer mot filer utføres ved kall til innebygde funksjoner. Her er en oppsummering over nyttige funksjoner, samt noen nye som ikke står i boka:

- `fopen()` åpner en fil.
- `fgets()` leser en linje fra en fil.
- `fwrite()` skriver en tekststreng til fil.
- `feof()` sjekker om slutten av filen er nådd.
- `file()` leser hele filen inn i en matrise, der hver linje blir et nytt matriseelement.
- `fclose()` lukker filen. Kan øke hurtigheten og redusere sannsynlighet for at filen overskrives av andre, men om en ikke bruker `fclose()` lukkes filen automatisk av PHP når scriptet har kjørt ferdig.
- `file_get_contents()` leser inn hele filen til en tekststreng.
- `file_put_contents()` lar deg skrive en tekststreng til fil, uten å først måtte åpne filen. Du kan angi med argumenter om innholdet skal overskrives eller legges til på slutten. Gjelder bare PHP 5 og nyere.
- `unlink()` sletter en fil.

Det fins også en rekke funksjoner for å behandle katagloer og hente ut filinformasjon.

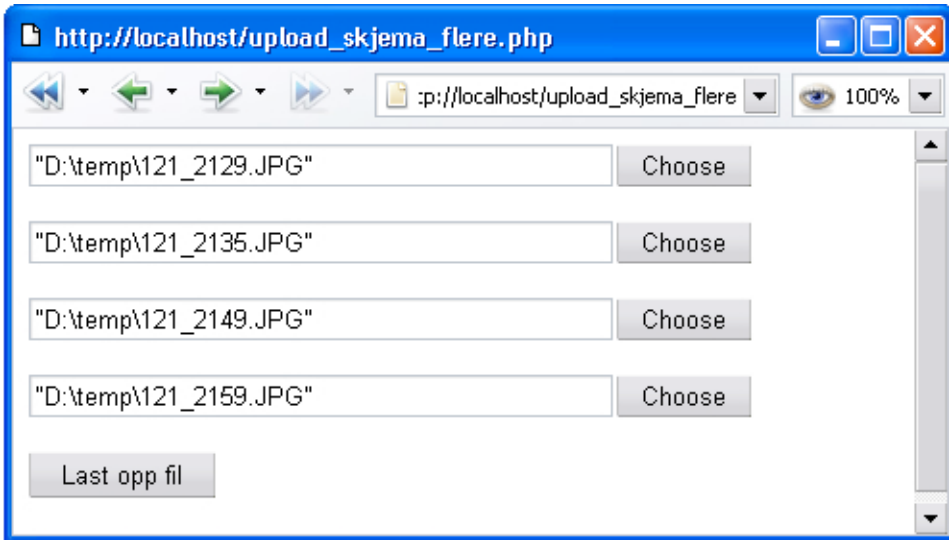
# 2. Opplasting av filer

Du kan også laste opp filer ved å spesifisere følgende skjema:

```
<form action='upload.php' method='post' enctype='multipart/form-data'>
<input type='file' name='filboks_navn' size='50'><p>
<input type='submit' value='Last opp fil' name='last_opp'>
</form>
```

Alle data angående den filen som velges, blir tilgjengelig i den superglobale matrisen `$_FILES[]`. Dersom du ønsker å laste opp flere filer, kan du velge om du vil ha unike navn på hver filboks, eller om du vil lage en matrise av alle elementene. En blanding er gjort her.

```
<form action='upload_alle.php' method='post' enctype='multipart/form-data'>
<input type='file' name='bildenr1' size='50'><p>
<input type='file' name='bildenr2' size='50'><p>
<input type='file' name='filboks_navn[]' size='50'><p>
<input type='file' name='filboks_navn[]' size='50'><p>
<input type='submit' value='Last opp fil' name='last_opp'>
</form>
```




Figur 1: For hver <input type='file'> kommer en ny tekstfelt med en velg-knapp til syne (Choose). Trykk på knappen "Last opp fil" (burde vært flertall) setter i gang et script som behandler filene valgt i hvert tekstfelt.

For å se hvilken matrisestruktur du når får å jobbe med videre, kan det være lurt å skrive ut følgende i et hjelpevindu / på ark:

```
<?php
echo "<pre>";
var_dump($_FILES);
?>
```

Her er et lite utdrag av resultatet. Filen som angitt i boks nummer 2 ble ikke riktig lastet opp, og derfor er det ikke satt noen type. I tillegg har elementet med nøkkel "error" fått verdi 1. Det skyldes at maksimalstørrelsen for filopplasting er satt til 2 MB, mens bildet 121_2135.JPG viser seg å være 2,2 MB stort. Denne maksimalverdien kan endres i direktivet `upload_max_filesize` i php.ini.

```
-------------8<----- klipp klipp ------8<-------
 ["bildenr2"]=>
 array(5) {
   ["name"]=>
   string(12) "121_2135.JPG"
   ["type"]=>
   string(0) ""
   ["tmp_name"]=>
   string(0) ""
   ["error"]=>
```

```
    int(1)
    ["size"]=>
    int(0)
}
-------------8<----- klipp klipp ------8<-------
```

# 3. Problem – to samtidige brukere

Boka gjennomgår hvordan en teller kan lages. Mange vil si at tellere var in på slutten av 90-tallet, men nå er utdatert. Uavhengig av om det er riktig, vil det ofte være veldig nyttig å kunne kartlegge trafikken på et nettsted eller på enkeltsider. Hva skjer om to eller flere brukere besøker siden nøyaktig samtidig? Det kan meget godt skje for litt større nettsteder, og scriptet med telleren vil ikke bli oppdatert riktig. Pseudokoden for å oppdatere telleren er:

**Oppdatere telleren – hvilke steg gjøres?**

```
1. Åpne fil i lesemodus
2. Les innhold fra fil inn i tellervariabel
3. Lukk fil
4. Oppdater tellervariabel med 1
5. Åpne fil i skrivemodus
6. Skriv tellervariabel tilbake til fil
7. Lukk fil
```

Betrakt en situasjon hvor to brukere forespør en side hvor tellerscriptet inngår nokså samtidig. Tjeneren vil dermed håndtere begge forespørslene hver for seg, og samtidig (hvorvidt prosessene kjøres parallelt eller at round-robin med CPU-tid skal vi ikke ta stilling til her). Vi kaller utføringen av den første forespørselen for F1, og den andre for F2.

Det interessante er hvordan F1 utføres i forhold til F2. Et mulig scenario der tiden går ovenfra og ned, er:

| **Tjener kommuniserer med** | **Kodelinje som utføres** | **Innhold i teller** | **Innhold på fil** |
|---|---|---|---|
| F1 | Linje nr 1, F1 | Teller_1 = 0 | 6 |
| F1 | Linje nr 2 og 3, F1 | Teller_1 = 6 | 6 |
| F1 og F2 | Linje nr 1, F2 | Teller_2 = 0 | 6 |
| F1 og F2 | Linje nr 4, F1 | Teller_1 = 7 | 6 |
| F1 og F2 | Linje nr 2 og 3, F2 | Teller_2 = 6 | 6 |
| F1 og F2 | Linje nr 5, F1 | Teller_1 = 7 | |
| F1 og F2 | Linje nr 4, F2 | Teller_2 = 7 | 6 |
| F1 og F2 | Linje nr 6 og 7, F1 | Teller_1 = 7 | 7 |
| F2 | Linje nr 5, F2 | Teller_2 = 7 | |
| F2 | Linje nr 6 og 7, F2 | Teller_2 = 7 | 7 |

F1 og F2 har to ulike tellervariabler, dette holder tjeneren rede på. Saken er den at siden de to forespørslene skjer nesten samtidig, vil både F1 og F2 lese verdien 6 fra fil, men uavhengig av hva F1 gjør er det F2 som har siste ordet i saken, siden F2 åpner filen før F1 har oppdatert verdien. Er du i tvil om dette virkelig kan skje? Prøv selv med følgende kode:

```
<?php /*script som øker sannsynligheten for to samtidige brukere /*
      //lese innhold fra filen
      $teller_fil = "teller.txt";
      if (   !( $filref = fopen($teller_fil, "r") )    )
           die ("En feil oppstod, får ikke åpnet filen");
      $teller = (int) fread($filref, 15);
      fclose($filref);

      $teller++;  //oppdaterer telleren
      echo "Du er besøkende nummer $teller";

      for($i=0; $i<6000000; $i++)
           ; //gjør ingenting fornuftig, men øker sjansen for feil

      //skriver tilbake oppdatert teller på samme fil.
      $filref = fopen($teller_fil, "w");
      fwrite($filref, $teller);

      //meget viktig at vi husker å lukke filen
      fclose($filref);
?>
```

For-løkken er satt inn i mellom pseudokodens linje 4 og 5 for å øke tidsforsinkelsen. Den går 6 millioner ganger og gjør absolutt ingenting – et semikolon angir en ”tom” kommando. Dette gjør at det blir et opphold på noen sekunder (avhengig av hvor rask tjener du har) for det tar tid å kjøre en løkke så mange ganger. I mellomtiden har du mulighet til å oppdatere to vinduer samtidig, manuelt. Dette simulerer to samtidige brukere og du ser at i stedet for å øke telleren i filen med 2, blir den bare økt med 1. Selv når for-løkken tas bort vil du kunne oppleve det samme. Situasjonen er selvsagt uholdbar, særlig dersom andre og mer viktige data enn tellere skal oppdateres.

Trikset for å unngå problemet er å låse filen ved åpning slik at bare én forespørsel kan operere mot filen samtidig. En fil låses og åpnes med funksjonen `flock()`, men med ulike konstanter som andre argument:

```
flock($filref, LOCK_EX);  //låse en fil for lesing og skriving
flock($filref, LOCK_UN);  //åpne en fil igjen
flock($filref, LOCK_SH);  //lov for flere å lese, men ikke skrive
```

Det er veldig viktig å huske å åpne en fil som er låst.

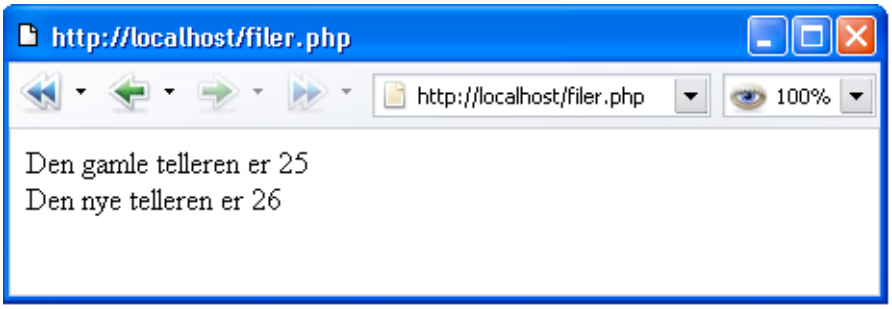
# 4. Kan filbehandling bli enklere?

Du ser av boka og eksempelt til nå at filbehandling krever litt planlegging. Det er helt bevisst at teller-eksempelet åpner filen to ganger i ulikt modus – nettopp for å illustrere hvordan operasjon mot filer fungerer. Det er derimot mulig å bruke mer hendige funksjoner som letter arbeidet. Som vist i oversikten fra punkt 7.3 i leksjonen er det to funksjoner som heter `file_get_contents()` og `file_put_contents()`. Den siste er bare tilgjengelig i

PHP 5 og nyere. Den første er nokså lik `file()`, men leser innholdet inn i en lang tekststreng. Linjeskift vil bli lagret som `\n`. Den siste tilsvarer operasjonene: `fopen()`, `fwrite()` og `fclose()`. I sum betyr det at telleren kan lages mye enklere enn det vi har gjort til nå:

```
<?php
      //smart måte å oppdatere en teller på, forutsetter PHP 5
      $teller = file_get_contents("teller.txt");
      echo "Den gamle telleren er $teller<br>";
      file_put_contents("teller.txt", ++$teller);
      echo "Den nye telleren er $teller";
?>
```

Dersom tallet 25 ligger lagret på fil i det brukeren kommer på besøk, vil dette bli presentert:




Figur 2: Tallet som ligger lagret på fil presenteres, oppdateres, og skrives tilbake til fil. Ulike brukere er dermed med på å oppdatere samme teller.

Legg merke til at telleren oppdateres **før** den skrives til fil i og med at det står `++$teller` inne i argumentet. Alternativt kunne en ha brukt en ekstra linje, slik:

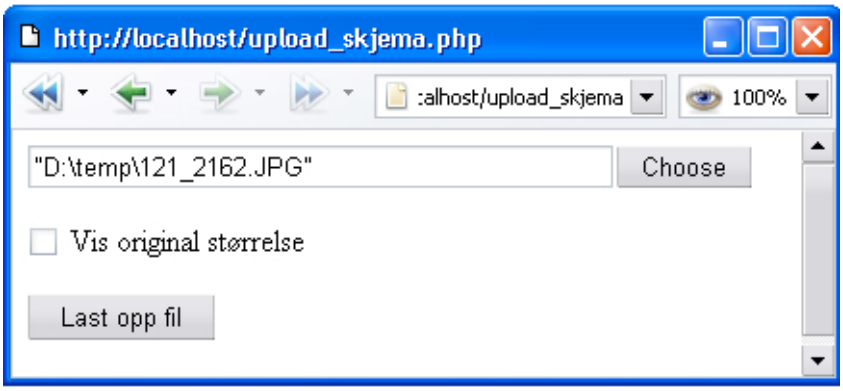
```
$teller++;
file_put_contents("teller.txt", $teller);
```

# 5. Last opp bilder uten å overskrive

Boka gjennomgår hvordan du kan laste opp filer, men samme filnavn som brukeren måtte ha på sin disk benyttes. Dette er ikke helt heldig, for en risikerer da at bildene blir overskrevet. Brukere kan angi rare filnavn, og det er lite heldig å bruke æ, ø og mellomrom etc i filnavn.

Ved å lage filnavnet på det opplastede bildet dynamisk hver gang slik at navnet blir på formen `bilde_1.jpg`, `bilde_2.jpg` og så videre unngår en at bilder får samme filnavn og overskrives. Det er i så måte lurt å kombinere selve opplastingen med en tekstfil som kun har en oppgave: Nemlig å holde orden på hvilken teller som neste gang skal brukes i filnavnet. Telleren lagres helst i klartekst som et tall. Etter å ha lest den inn må den brukes for å lage riktig bildenavn.

Vi lager først et skjema med opplastingsmulighet (en fil) og en avkrysningsboks for å angi om bildet skal vises i full eller redusert størrelse når opplastingsoperasjonen oppsummeres.




Figur 3: Opplastingsfelt og radioknapp (Vis original størrelse).

Opplastingsknappen har navnet ”filboks_navn”. Avkrysningsknappen heter ”stor”. Her er en kodbit som kan brukes til å laste opp filen og vise bildet i riktig størrelse. Navnet blir unikt. Dette forutsetter at katalogen **opplastet/** er skrivbar og har en skrivbar fil som heter **nr.txt**. Denne filen har tallet 0 i seg i utgangspunktet.

```
<?php
//$temp_fil er et midlertidig navn bestemt i php.ini
$temp_fil = $_FILES['filboks_navn']['tmp_name'];

//henter ut og oppdaterer tellerverdi
$filnavn = "opplastet/nr.txt";
$teller = file_get_contents($filnavn);
file_put_contents($filnavn, ++$teller); //betyr $teller = $teller + 1

//mimetypen til filen, typisk på formen "image/gif"
$filtype = $_FILES['filboks_navn']['type'];
if ( strstr($filtype, "jpeg") )  //image/jpeg
      $etternavn = ".jpg";
elseif ( strstr($filtype, "gif") ) //image/gif
      $etternavn = ".gif";
elseif ( strstr($filtype, "png") )//image/png
      $etternavn = ".png";
else
      $etternavn = ".ukjent";

//lager navnet bilde_3 hvis forrige bilde var bilde_2
$nyttFilnavn = "opplastet/bilde_" . $teller . $etternavn;
copy($temp_fil, $nyttFilnavn)
      or die ("Feil, e-post med rapport er sendt til webmaster");

echo "<h3>Dette er bildet som ble lastet opp</h3>";
if ( isset($_POST['stor']) ) {
      $bildeInfo = getimagesize($nyttFilnavn);
      $dimensjoner = $bildeInfo[3];
}
else $dimensjoner = "height='200'";
echo "<img src='$nyttFilnavn' $dimensjoner>";
?>
```

Etternavnet blir laget basert på mime-typen til bildet. Det er nemlig ikke endelsen som angir hvorvidt bildet er et bilde eller ikke. En bruker kan godt endre filnavnet til et bilde til ettellerannet.txt og fortsatt vise dette som et bilde så sant mimetypen er image/jpeg.

Legg spesielt merke til de tre setningene i fet skrift. Den første har vi allerede gjennomgått. Den andre setter sammen de ulike delene av filnavnet slik at korrekt nummerering brukes. Deretter kan bildet kopieres fra temporær katalog til riktig katalog.

Den tredje setningen i fet skrift kaller funksjonen getimagesize(). Denne funksjonen er hendig, fordi den returnerer ulik informasjon om et bilde i en matrise. I elementet med nøkkel 0 ligger bredden på bildet i antall piksler, nøkkel 1 angir høyden, nøkkel 2 et flagg som angir typen (se manualen). Elementet med nøkkel 3 har en streng med verdien `height="yyy" width="xxx"` og kan dermed brukes til å vise bildet i sin originale størrelse.

# 6. Om logging

Alle webtjenere kan logge informasjon. Ved bruk av Apache ligger loggene under katalogen logs/ der hvor Apache er installert. Se gjerne på filen access.log, og merk følgende:

- Apache logger *alle* forespørsler som mottas.
- Loggen har en linje for hver forespørsel som er mottatt.
- Legg merke til filstørrelsen på loggen. Hvis du har programmert, testet og feilet en del, vil du se at loggen er stor.
- Loggen kan åpnes i en tekstbehandler.
- Koden for å sette inn et bilde i HTML er `<img src='bildenavn.jpg'>`, hvor en også kan referere til bilder i andre kataloger eller andre steder på Internett.
- Når en side har 100 bilder, vil det komme 101 innslag i loggen. Dette skyldes at når klienten forespør en tjener om å få en side, mottar klienten en HTML-kildekoden i retur. Denne tolkes, og for hver img-tag (samt andre tagger med referanse til eksterne ressurser, som stilsett, JavaScript etc) sender nettleseren en ny forespørsel etter ønsket ressurs. Denne logges på lik linje med forespørselen etter en nettside. I praksis kan nettleseren også forespørre andre ressurser, som en zip-fil, en PDF-fil etc. Dermed kan du lenke direkte til slike filer med HTML-kode.

Du kan se hva som skjer ved besøk på for eksempel Adresseavisen (www.adressa.no) fra nettlesere som viser statusinformasjon etter hvert som sidens elementer lastes ned. Opera er et godt eksempel på en slik nettleser.

Du ser nå at mye informasjonen lagres i Apaches logger. Det kan være nødvendig å lage din egen logging. Kanskje har du ikke tilgang til webtjenerens logger (hvis du bruker webhotell eller liknende). Webtjeneren logger også mye informasjon og har ofte mye besøk. Dermed blir loggfilene fort store – veldig store. Mange webtjenere er derfor satt opp slik at loggene overskrives (eventuelt biter seg selv i halen) etter en viss periode. Det er surt å be systemansvarlig om å få logger bare for å få vite at på grunn av en maksimumsstørrelse på 100 MB store loggfiler, eksisterer det loggdata for 14 dager tilbake i tid, ikke mer… Logger du selv har du mye større kontroll. Pass på størrelsen på tekstfilene og vær klar over at logging kan skape en reduksjon i responstid for brukeren!