LinuxDoc+Emacs+Ispell-HOWTO

Författare: Philippe MARTIN (feloy@wanadoo.fr)
Engelsk översättare: Sébastien Blondeel (Sebastien.Blondeel@lifl.fr)
Svensk översättare: Linus Åkerlund (uxm165t@tninet.se) v0.4, 27 February 1998. Svensk översättning: 10 juni 1998.

Detta dokument är riktat till författare och översättare av Linux-HOWTOn eller andra dokument för The Linux Documentation Project. Det ger dem ledtrådar om hur de ska använda verktyg såsom Emacs och Ispell.

Innehåll

1 Inledning

1.1 Upphovsrätt

1.2 Tillkännagivanden

Ett speciellt tack går till Sébastien Blondeel, som är en hemsk typ, och frågade mig en massa om att konfigurera Emacs. Hans intelligenta frågor har gjort att jag förstår det bättre, och vidarebefordrar denna kunskap till dig, genom detta dokument.

1.3 Kommentarer

Tveka inte att tala om för mig, om det är något du tycker skulle kunna göra detta dokument bättre. Jag kommer tänka över din kritik noggrant.
Tveka ej heller att ställa frågor, relaterade till de ämnen som här diskuteras, till mig. Jag besvarar dem mer än gärna, eftersom de hjälper mig att vidare förbättra detta dokument. [1]

1.4 Versioner

Detta dokument behandlar följande versioner:

- Sgml-tools version 0.99,
- Emacs version 19.34,
- Ispell version 3.1,
- Alla Emacs-bibliotek, vilka det refereras till i detta dokument, distribueras med ovan nämnda version av Emacs, förutom `iso-sgml`, vilket distribueras med XEmacs, och `psgml`, vilket är ett fristående bibliotek.

[1] Översättarens anmärkning: Om engelskan är ful, tja, det är jag också!

1.5 Svenska översättarens anmärkningar

Detta dokument är en översättning av den engelska versionen. Originalet är skrivet på franska, ett språk jag inte behärskar. Jag hoppas att jag lyckats bevara all information som finns med i originaldokumentet. Som vanligt är alla välkomna att komma med anmärkningar och kritik. Ändra gärna i SGML-filen och skicka de korrigerade versionerna till mig, på `uxm165t@tninet.se`.

2 Introduktion

2.1 SGML

Standard Generalised Mark-up Language, eller **SGML**, är ett språk för att definiera dokument-typer.

Man kan t.ex. definiera dokument-typen *recept*, med en första del för att presentera ingredienserna, en andra del för att presentera hjälpmedlen, en tredje del vilken ger stegvisa instruktioner om hur tårtan ska bakas och en trevlig, avslutande bild, för att visa resultatet av allt detta.

Detta kallas en *definition av en dokument-typ* (Document type Definition. övers.anm.). Den definierar inte hur den slutliga produkten kommer se ut, den definierar bara vad den kommer att innehålla.

För att använda samma exempel igen; jag är säker på att du, då du läser om min idé om ett recept, känner igen dina, eller din favoritkocks. De ser trots allt olika ut: mina har en bild i det övre vänstra hörnet av badrumsskåpet, och listan över ingredienser finns på bakgården, mellan swimmingpoolen och grillen. Dina?

Tack vare standarddefinitionen, så kan man skriva ett dokument utan att ta hänsyn till hur det slutligen kommer se ut, för läsaren.

2.2 `LinuxDocs` typ-definition

Denna typ används, som du kanske har kunnat gissa dig till, till att skriva dokument vilka är relaterade till Linux.

Sådana dokument är typiskt uppbyggda som följer: de börjar med en titel, följd av författarens namn, versions-numret och datumet. Sedan kommer sammanfattningen (så att du inte behöver bläddra igenom hela alltet, innan du inser att det inte alls var vad du var ute efter), sedan innehållsförteckningen, vilken visar strukturen, så att de som har bråttom kan hoppa direkt till det avsnitt de vill läsa.

Sedan följer en lista över kapitel, avsnitt och stycken. Bland dessa kan man infoga delar av program, ändra typsnitt, för att framhäva ett ord eller en mening, infoga listor, referera till andra delar av dokumentet osv.

För att skriva ett sådant dokument behöver du bara, vid rätt tidpunkt, specificera titeln, författaren, datumet och dokument-versionen, kapitlen och avsnitten, när en lista ska infogas, vilka dess element är osv.

2.3 `SGML-Tools`

SGML-Tools omvandlar ett dokuments specifikationer till det slutgiltiga resultatet, i det format du önskar. Om du vill ha det i ditt personliga bibliotek, väljer du *PostScript*. Om du vill dela det med världen, genom webben, så blir det *HTML*. Om du inte kan hjälpa det, och måste läsa det under Windows, så kan du omvandla det till *RTF*, så att det kan läsas av vilken ordbehandlare som helst. Eller så kanske du använder alla tre formaten, för att anpassa det till ditt växlande humör.

SGML-Tools är tillgängligt via anonym FTP från `ftp://ftp.funet.fi/public/ftp/pub/linux/utils/text/`

3 Ditt första dokument

3.1 Från ett textdokument

Om du vill gör om ett textdokument till SGML, för att omvandla det till andra format, ska du göra på följande sätt:

1. Lägg till de följande raderna i början:

```
<!doctype linuxdoc system>
 <article>
  <title>Här ska titeln vara</title>
  <author>
  författarens namn, e-post-adress osv.
  </author>
  <date>
  version och datum
  </date>
```

2. Om du kort beskriver dokumentets innehåll i början, innefatta stycket i `<abstract>`- och `</abstract>`-taggarna.
3. Infoga sedan `<toc>`-taggen, vilket står för *Table Of Contents* (innehållsförteckning. övers.anm.).
4. Ersätt, i början av varje kapitel, raden som anger nummer och titel för kapitlet med:

   ```
   <sect>Kapitlets titel
   ```

   och lägg till `</sect>`-taggen i slutet av kapitlet.

   Observera : Du behöver inte ange kapitel-nummer, detta görs automatiskt.
5. Fortsätt på samma sätt med avsnitten. Du måste ta bort numren och ”tagga” deras titlar med `<sect1>`, och avsluta dem med `</sect1>`.
6. Du kan även, på ett liknande sätt, ange så många som 4 nästade nivåer i avsnitten, genom att använda `<sectn>`- och `</sectn>`-taggarna, där `n=` 2, 3 eller 4.
7. Lägg, i början av varje stycke, in `<p>`-taggen.
8. Om du behöver framhäva vissa delar, ”tagga” dem med `<it>` och `</it>` (*kursiverat*), `<bf>` och `</bf>` (**fetstil**) eller `<tt>` och `</tt>` (`skrivmaskins-stil`).
9. För att infoga en lista, som den följande:

   ```
   Detta är en lista om fyra rader:

     - här är första raden
     - sedan kommer andra raden
     - ännu en
     - det var det.
   ```

så måste du ersätta den med:

```
Detta är en lista om fyra rader:
<itemize>
<item>här är första raden
<item>sedan kommer andra raden
<item>ännu en
<item>det var det.
</itemize>
```

10. När ett helt block är en del av ett program, eller något annat som måste sticka ut:

```
<verb>
 10 REM Herre gud, vad är det här?
 20 REM Jag trodde det här hade försvunnit för länge sedan!
 30 PRINT "Jag är tillbaks för att";
 40 PRINT "rädda världen."
 50 INPUT "Från vem, menar du? ",M$
 60 IF M$="Bill" THEN PRINT "Du är vis.":GOTO PARADISET
 70 ELSE PRINT "Du har inte hajat något...":GOTO RIKMOND
</verb>
```

11. På detta stadium är dina kunskaper i SGML-formattering ganska hyffsade. Om du vill förfina ditt dokument, så kan du ta en titt på användar-manualen till **SGML-Tools**, vilken ger mer detaljer om dokument-typen **LinuxDoc**.

4 Konfigurera Emacs

4.1 Tecken med accent

Om du vill skriva dokument på franska, eller något annat västeuropeiskt språk, så behöver du 8-bitars tecken. Så här ställer du in Emacs för att acceptera sådana tecken.

4.1.1 Att visa 8-bitars tecken

För att få Emacs att visa 8-bitars tecken, så måste du ha följande rader i din `.emacs`-fil:

```
(standard-display-european 1)
(load-library "iso-syntax")
```

Om du använder Emacs på en terminal, som har stöd för 8-bitars tecken, så kan du använda `iso-ascii`-biblioteket (`(load-library "iso-ascii")`), vilket säger åt Emacs att visa sådana tecken, så bra den kan.

4.1.2 Att skriva 8-bitars tecken

Om ditt tangentbord låter dig skriva tecken med accenter, så är det inget problem. Om inte, så kommer här några hjälpmedel:

`iso-acc`-biblioteket Emacs `iso-acc`-bibliotek låter dig skriva 8-bitars tecken på ett 7-bitars tangentbord. För att använda det, infoga följande i din `.emacs`-fil:

```
(load-library "iso-acc")
```

När du sedan kör Emacs och öppnar en fil att editera, skriv `Meta-x iso-accents-mode`.

Du kan skriva **é**-tecknet, i det svenska ordet *kafé*, genom att skriva `'` och sedan `e`. Mer generellt så kommer du skriva ett tecken med accent genom att först skriva accenten, och sedan bokstaven som accenten ska finnas till (stora eller små bokstäver). De följande är accenter du kan vilja använda:

- `'` : Höger-lutad (akut) accent
- `` ` `` : Vänster-lutad (grav) accent
- `^` : Cirkumflex
- `"` : Punkter
- `~` : Tilde och andra specialfall (cf iso-acc.el).
- `/` : För att få ett streck över ordet

Om du behöver något av dessa tecken, och inte en bokstav med accent, tryck mellanslag efter tecknet. För att skriva t.ex. *'kafé'*, skriv `'` ⟨*mellanslag*⟩ `k a f ' e '`.

Du hittar alla möjliga kombinationer i `iso-acc.el`-filen.

⟨*Meta*⟩-tangenten Vissa terminaler låter dig skriva 8-bitars tecken med ⟨*Meta*⟩-tangenten (eller ⟨*Alt*⟩). Om du t.ex. trycker ⟨*Meta*⟩**-i**, så får du **é**-tecknet.

Men Emacs reserverar ⟨*Meta*⟩-tangenten för andra ändamål, och jag känner inte till något bibliotek som låter dig använda det för tecken med accent.

Detta är en lösning:

```
(global-set-key "\ei" '(lambda () (interactive) (insert ?\351)))
                  -                                        ---
```

En sådan rad, om den infogas i din `.emacs`-fil, låter dig skriva **é** genom att använda ⟨*Meta*⟩**-i**. Du kan omdefiniera de kombinationer du behöver, genom att byta ut **i** mot den rätta tangenten och **351** med den rätta koden (koden finner du i tecken-uppsättningen ISO-8859-1).

Varning! Vissa lokala lägen kan omdefiniera sådana tangent-kombinationer.

4.1.3 Att visa 8-bitars SGML-tecken

Under SGML kan du skriva tecken med accent med makron. För att skriva t.ex. **é**-tecknet, så skriver du **é**. Rent generellt så kan de applikationer som behöver kunna läsa SGML redan läsa 8-bitars tecken, så dessa makron behövs inte. Men vissa kan inte det. Eftersom det finns ett sätt att lösa problemet, så vore det onödigt att inte använda makrona.

`iso-sgml`-biblioteket låter dig skriva tecken med accent under Emacs, som alltid, men då du sparar din fil på disk, så omvandlas dessa 8-bitars tecken till sina motsvarigheter i SGML.

På grund av detta bibliotek är det alltså lätt att skriva och läsa dina dokument under Emacs, och du kan vara säker på att applikationer som inte klarar av 8-bitars tecken kommer att acceptera dina dokument.

För att använda detta bibliotek behöver du bara lägga till följande rader i din `.emacs`-fil:

```
(setq sgml-mode-hook
'(lambda () "Defaults for SGML mode."
 (load-library "iso-sgml")))
```

4.2 SGML-läge

Då du laddar en fil med **.sgml**-ändelsen, så går Emacs automatiskt in i **sgml-läge** (sgml mode). Om det inte gör det, så kan du manuellt säga åt det att göra det genom att skriva `Meta-x sgml-mode`, eller automatiskt, genom att lägga till följande rader till din `.emacs`-fil:

```
(setq auto-mode-alist
(append '(("\.sgml$"  . sgml-mode))
          auto-mode-alist))
```

Detta läge låter dig t.ex. välja hur du ska infoga 8-bitars tecken. Med `Meta-x sgml-name-8bit-mode` (eller meny-alternativet *SGML/Toggle 8-bit insertion*), kan du välja att skriva in 8-bitars tecken som de är, eller i SGML-form, alltså i formen **&...;**.

Det låter dig även dölja eller visa SGML-taggarna, med `Meta-x sgml-tags-invisible` (eller meny-alternativet *SGML/Toggle Tag Visibility*).

4.3 PSGML-läge

PSGML-läget (PSGML mode) är väldigt hjälpsamt, om du editerar SGML-dokument med Emacs.

`psgml-linuxdoc`-dokumentationen förklarar hur du ska installera detta läge och använda det med *LinuxDoc*.

4.4 Diverse annat

4.4.1 auto-fill-läge

I det normala läget måste du använda ⟨*Return*⟩, då du kommer till slutet av raden, för att komma till nästa rad, annars kommer raden att ta upp hela stycket. Om du använder ⟨*Return*⟩ för att komma till nästa rad, så får du ett stycke med ojämna kanter.

Om du låter vissa rader gå bortom den rimliga längden, så kommer du inte kunna se dem i vissa editorer.

auto-fill-läget automatiserar detta jobbiga arbete: då du går bortom en viss kolumn (den 70nde, som standard), så tas du automatiskt till nästa rad.

Så här slår du på detta läge, och anger radlängden till 80 tecken:

```
(setq sgml-mode-hook
      '(lambda () "Defaults for SGML mode."
```

```
(auto-fill-mode)
(setq fill-column 80)))
```

5 Ispell

Om du vill stavningskontrollera dina dokument från Emacs, så kan du använda **Ispell**-paketet och dess Emacs-läge.

5.1 Att välja standardordlistor

Du kan ställa in Emacs så att det, då du laddar en fil, automatiskt väljer vilka ordlistor det ska använda (det kan använda flera). Den första, och säkerligen den viktigaste, är huvudordlistan, vilken distribueras med Ispell. Du kan välja mellan flera olika språk. Den andra är din personliga ordlista, där Ispell infogar ord som det inte kunde finna i huvudordlistan, men du sade åt det att komma ihåg.

En svensk (och även en dansk) ordlista kan hämtas på `http://www.sslug.dk/ispell`.

Om du vill använda den standardiserade franska ordlistan som kommer med Ispell, och om du vill använda filen `.ispell-perso-ord`, i din hemkatalog, som din personliga ordlista, så lägg till följande rader till din `.emacs`-fil:

```
(setq sgml-mode-hook
'(lambda () "Defaults for SGML mode."
(setq ispell-personal-dictionary "~/.ispell-perso-ord")
(ispell-change-dictionary "francais")
))
```

5.2 Välja speciella ordlistor för vissa filer

Du kan få problem om du inte stavningskontrollerar alla dokument i samma språk hela tiden. Om du översätter dokument, så är det ganska troligt att du byter språk (och ordlistor) väldigt ofta.

Jag känner inte till något Lisp-sätt att välja, antingen automatiskt eller med ett enkelt mus-klick, huvudordlista eller personliga ordlista, som ska associeras med språket du för tillfället använder. (Om du känner till något om detta, tala om det för mig!)

Det är dock möjligt att ange, på slutet av filen, vilka ordlistor du vill använda för den aktuella filen (och endast denna). Det räcker att lägga till dem som kommentarer, så att Ispell kan läsa dem då det kör stavningskontrollen (spell-check).

```
<!-- Local IspellDict: english -->
<!-- Local IspellPersDict: ~/emacs/.ispell-english -->
```

Om du redan, i din `.emacs`-fil, har angivit att dina standard-ordlistor är de franska, så kan du lägga till dessa rader i slutet av varje engelsk fil.

5.3 Stavningskontrollera dina dokument

För att stavningskontrollera ett helt dokument, använd, varifrån som helst i dokumentet, `Meta-x ispell-buffer`-kommandot. Du kan även välja att bara kontrollera en viss del av dokumentet:

- Markera början av avsnittet med `Ctrl-Spc` (mark-set-command),
- Gå till slutet av avsnittet du vill kontrollera,
- Skriv `Meta-x ispell-region`.

Sedan kör Emacs Ispell. Då det stöter på ett okänt ord visar det dig det aktuella ordet (vanligtvis markerat) och ber dig välja mellan:

- **mellanslag** behåll ordet, endast denna gång,
- **i** behåll ordet, och infoga det i din personliga ordlista,
- **a** behåller ordet för denna session,
- **A** behåller ordet i denna fil, och infogar det i filens lokala ordlista,
- **r** låter dig rätta till ordet för hand,
- **R** låter dig rätta till alla instanser av det felstavade ordet,
- **x** avbryter kontrollen och placerar markören där den var från början,
- **X** avbryter kontrollen och lämnar markören där den är, vilket låter dig korrigera filen; du kan senare fortsätta kontrollen om du skriver `Meta-x ispell-continue`,
- **?** ger dig hjälp.

Om ispell finner ett eller flera ord som liknar det okända ordet, så visar det dem i ett litet fönster, vart och ett med en siffra framför sig. Tryck bara på motsvarande siffertangent, för att ersätta det felstavade ordet med det korresponderande ordet.

5.4 Personlig ordlista kontra filens lokala ordlista

i-tangenten låter dig infoga ett ord i din personliga ordlista, och **A** låter dig infoga ett ord i en lokal ordlista för den aktuella filen.

Den lokala filens ordlista är en sekvens av ord infogade i slutet av filen, som kommentarer, vilka läses in a Ispell varje gång det ska kontrollera filen. På det sättet kan du behålla vissa ord, vilka är acceptabla i denna fil, även om de inte med nödvändighet är acceptabla i andra filer.

Enligt min åsikt så är det bättre att reservera den personliga ordlistan för ord som huvudordlistan inte känner till, men som tillhör språket (som sammansatta ord), plus några vanliga ord som substantiv och andra ord (som *Linux*), om de inte är alltför lika ett riktigt ord, som finns i huvudordlistan; att lägga till alltför många ord i den personliga ordlistan, såsom förnamn, kan vara farligt, eftersom de kan likna ett ord som ingår i språket (man skulle kunna tänka sig att Ispell blir förvirrat av det följande: '*Det roligaste med Young Ones var när Neil skulle koka linus-soppa.*'[2]

[2]Om jag skulle lagt in *Linus* i min personliga ordlista!

5.5 Direkt stavningskontroll

Ispell kan utföra stavningskontroll på filen medan du skriver. Till detta måste du använda **ispell-minor-mode**. För ett starta eller avsluta det, skriv `Meta-x ispell-minor-mode`. Ispell *piper* varje gång du skriver ett ord som det inte känner igen.

Om dessa *pip* irriterar dig (eller om din rumskamrat tar en tupplur), så kan du ersätta de irriterande *pipen* med att skärmen blinkar till, vilket du gör med kommandot `Meta-x set-variable RET visible-bell RET t RET`. För att för evigt tysta Emacs, kan du lägga till följande rad i din `.emacs`-fil:

```
(setq visible-bell t)
```

6 Fula knep

6.1 Att automatiskt infoga ett dokumenthuvud

Emacs låter dig *hook*-a vissa funktioner till händelser (öppna en fil, spara, köra igång ett nytt läge osv).

autoinsert-biblioteket använder denna funktion: när du öppnar en ny fil under Emacs, så infogar detta bibliotek, i enlighet med filtypen, ett *standardiserat* dokumenthuvud.

I vårt fall skulle detta *standardiserade* huvud kunna vara den del som deklarerar dokument-typen (LinuxDoc), titeln, författaren och datumet.

Jag ska beskriva två sätt att infoga ett sådant huvud. Du skulle kunna infoga en förpreparerad fil, vilken innehåller denna information, eller så kan du köra en **elisp**-rutin.

6.1.1 Genom att infoga en fil

Först måste du säga åt Emacs att köra `auto-insert`-läget då en fil öppnas, att sedan läsa **autoinsert**-biblioteket, vilket deklarerar `auto-insert-alist`-listan, vilken vi måste ändra i. Denna lista anger huvudet som ska infogas för varje filtyp. Som standard måste filen som ska infogas finnas i `~/insert/`-katalogen, men det är möjligt att ändra `auto-insert-directory`-variabeln, om du vill lägga den någon annanstans.

Lägg till följande rader till din `.emacs`-fil, för att infoga filen `~/emacs/sgml-insert.sgml`, varje gång du öppnar en ny SGML-fil:

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(setq auto-insert-directory "~/emacs/")
(setq auto-insert-alist
      (append '((sgml-mode .  "sgml-insert.sgml"))
              auto-insert-alist))
```

Sedan kan du skriva in ditt specialkomponerade huvud i filen `~/emacs/sgml-insert.sgml`, starta om Emacs och öppna t.ex. `foobar.sgml`-filen. Emacs ska då be dig bekräfta att filen ska infogas automatiskt, och om du svarar ja så ska filen infogas.

6.1.2 Genom att köra en rutin

Detta fungerar som förut, men istället för att sätta `auto-insert-alist` till en fil att infoga, så måste du sätta den till en funktion att köra. Så här ska du göra, om du vill skriva denna funktion i en fil som heter `~/emacs/sgml-header.el` (det finns ingen anledning att betunga din `.emacs`-fil med en sådan funktion, eftersom den kan bli ganska lång):

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(add-to-list 'load-path "~/emacs")
(load-library "sgml-header")
(setq auto-insert-alist
      (append '(((sgml-mode .  "SGML Mode") . insert-sgml-header))
              auto-insert-alist))
```

I A (appendixet) hittar du ett exempel på en `insert-sgml-header`-funktion.

A En insert-sgml-header-funktion

Denna funktion låter användaren infoga ett specialkomponerat dokumenthuvud i ett Linux Documentation Project-dokument. Den kan anropas automatiskt när en fil öppnas, eller explicit, av användaren.

Denna funktion frågar användaren, genom *mini-bufferten*, om en del information, av vilken en del är nödvändig, och annat ej.

Först kommer titeln. Om ingen anges, så avslutas funktionen omedelbart, och infogar inget. Sedan kommer datumet, författaren, hans e-postadress och hemsida (de två sista är valfria).

Sedan kommer en fråga efter namnet på översättaren. Om det inte finns någon, tryck bara *Return* och inga fler frågor om en hypotetisk översättare kommer ställas. Om det finns en, så tillfrågas du om hans e-postadress och hemsida (också valfria).

Funktionen skriver sedan in detta i den aktuella buffern, inklusive informationen du skrev in i en mall, och inklusive taggarna, som används till sammanfattning och första kapitlet. Slutligen placeras markören där sammanfattningen ska skrivas.

```
(defun insert-sgml-header ()
  "Infoga dokumenthuvudet för ett LinuxDoc-dokument"
  (interactive)
  (let (titel forfattare epost hem oversattare epost-oversattare hem-oversattare datum
             starting-point)
    (setq titel (read-from-minibuffer "Titel: "))
    (if (> (length titel) 0)
        (progn
          (setq datum (read-from-minibuffer "Datum: ")
                forfattare (read-from-minibuffer "Författare: ")
                epost (read-from-minibuffer "Författarens e-post: ")
                hem (read-from-minibuffer "Författarens hemsida: http://")
                oversattare (read-from-minibuffer "Översättare: "))
          (insert "<!doctype linuxdoc system>\n<article>\n<titel>")
          (insert titel)
```

```
(insert "</titel>\n<forfattare>\nFörfattare: ") (insert forfattare) (insert "<newline>\n")
(if (> (length epost) 0)
    (progn
      (insert "<htmlurl url=\"mailto:")
      (insert epost) (insert "\" name=\"") (insert epost)
      (insert "\"><newline>\n")))
(if (> (length hem) 0)
    (progn
      (insert "<htmlurl url=\"http://")
      (insert hem) (insert "\" name=\"") (insert hem)
      (insert "\">\n<newline>")))
(if (> (length oversattare) 0)
    (progn
      (setq epost-oversattare (read-from-minibuffer "Översättarens e-post: ")
            hem-oversattare (read-from-minibuffer "Översättarens hemsida: http://"))
      (insert "Översättare : ")
      (insert oversattare)
      (insert "<newline>\n")
      (if (> (length epost-oversattare) 0)
          (progn
            (insert "<htmlurl url=\"mailto:")
            (insert epost-oversattare) (insert "\" name=\"")
            (insert epost-oversattare)
            (insert "\"><newline>\n")))
      (if (> (length hem-oversattare) 0)
          (progn
            (insert "<htmlurl url=\"http://")
            (insert hem-oversattare) (insert "\" name=\"")
            (insert hem-oversattare)
            (insert "\"><newline>\n")))))
(insert "</forfattare>\n<datum>\n")
(insert datum)
(insert "\n</datum>\n\n<abstract>\n")
(setq point-beginning (point))
(insert "\n</abstract>\n<toc>\n\n<sect>\n<p>\n\n\n</sect>\n\n</article>\n")
(goto-char point-beginning)
))))
```