

# Emacs: Introduktion og C Guide

Alexander Christensen

August 4, 2017

## Abstract

Introduktion til Emacs, samt guide til opsætning af C udviklingsmiljø i Emacs; herunder indentering, kompilering, auto-completion, og konfiguration af pakker.

## Contents

<b>1</b>	<b>Introduktion</b>	<b>1</b>
<b>2</b>	<b>Notation</b>	<b>2</b>
<b>3</b>	<b>Init-filen</b>	<b>2</b>
<b>4</b>	<b>Pakker</b>	<b>3</b>
<b>5</b>	<b>Customization</b>	<b>3</b>
<b>6</b>	<b>Ielm</b>	<b>3</b>
<b>7</b>	<b>Hooks</b>	<b>4</b>
<b>8</b>	<b>Automatisk backup</b>	<b>5</b>
<b>9</b>	<b>Indentering</b>	<b>6</b>
<b>10</b>	<b>Kompilering</b>	<b>7</b>
<b>11</b>	<b>Ekstra</b>	<b>8</b>

## 1 Introduktion

Denne guide henvender sig primært til studerende på DIKU, som skal til at påbegynde deres andet studieår, og som derfor skal lære at programmere C. Der vil blive lagt væk på pakker og konfigurationer, der knytter sig specifikt til C (eller C++) programmering, og mange fundamentale elementer som fx navigation mellem filer, genvejstaster osv. vil ikke blive berørt.

## 2 Notation

I Emacs jargon betyder "**C-c**" at man trykker "Control c" på sit tastatur, "**M-c**" betyder "Alt c", og "**RET**" betyder "enter". En **buffer** er en åben instans af en fil, en **frame** er et programvindue, og en **frame** kan inddeles i flere **windows** / **vinduer**, som er pladsholdere for **buffere**. Cursoren kaldes for **point**.

## 3 Init-filen

Kernen omkring Emacs er den utroligt fleksible init-fil. Hvis Emacs åbnes direkte fra en 'frisk' installation ligner det noget, man vil holde sig langt væk fra! Det er meningen, at Emacs skal have en masse pakker installeret, og en rutineret init-fil kommer let op på flere tusind linjer (den kan deles op over flere filer!).

Så hvor finder man sin init-fil? Svaret afhænger af styresystemet. Flere detaljer kan ses på denne hjemmeside: <https://www.emacswiki.org/emacs/InitFile>.

### nyttig info:

"**C-x C-e**" i slutningen af en linje eller blok med E-lisp kode vil evaluere denne linje eller blok.

"**M-x**" → "**eval-buffer**" → "<RET>" vil læse hele init-filen igennem og aktivere (evaluere) alle skrevne konfigurationer. I visse tilfælde er det dog nødvendigt at lukke Emacs og starte det op igen for at se ændringerne - det sker heldigvis ikke særlig ofte.

Efter ændringerne i init-filen er evalueret, kan de aktiveres i eksempelvis en C buffer ved at skrive "**M-X**" → "**c-mode**" → "<RET>".

## 4 Pakker

Pakker til Emacs kan installeres ved at trykke "M-x" → "list-packages". Der kan søges efter pakker med "C-s" eller "C-r", og hvis point er placeret i starten af linjen for en pakke, kan denne pakke markeres til installation ved at trykke "i", eller markeres til sletning ved at trykke "d". For at udføre alle markerede installationer/afinstallationer tryk "x".

For at få mere udbytte af den indbyggede pakke-manager (læs: flere tilgængelige pakker), da kan der med fordel tilføjes **Melpa**-arkivet til init-filen, som følgende:

---

```
elisp - init.el
;; Add and enable the MELPA package archive
(require 'package)
(add-to-list 'package-archives
  '("melpa" . "http://melpa.milkbox.net/packages/") t)

; activate all the packages (in particular autoloads)
(package-initialize)

; fetch the list of packages available
(unless package-archive-contents
  (package-refresh-contents))
```

---

Disse indstillinger bør stå allerøverst oppe i init-filen.

## 5 Customization

I Emacs er der ingen grænser for, hvad der kan ændres på. Et glimrende eksempel er hvis man skriver "M-x" → "customize-face" → "all faces" → "<RET>": Så kommer der en lang liste frem over alle faces (dvs. farver, skrifttype/størrelser, osv), der kan konfigureres. I Emacs er der ikke noget, der ikke kan konfigureres, og det er ganske sjovt at 'lege med farverne'.

- I stedet for "all faces" kan man også vælge "default": så kan man ændre baggrundsfarven i Emacs.

- Alle customizations man laver bliver automatisk gemt i init-filen når man trykker 'Save for future sessions'.

## 6 Ielm

"Inferior Emacs Lisp Mode". Her kan man afprøve sproget **elisp**, som init-filen er skrevet i. Aktiveres ved at skrive "M-x" → "ielm" → "<RET>". Det er en god øvelse, hvis man gerne vil lære at skrive sine egne udvidelser. (det får man brug for, hvis man har anvendt Emacs længe nok!)

```
Emacs - *ielm*
*** Welcome to IELM *** Type (describe-mode) for help.
ELISP> (defun my-awesome-function (coffee-in-blood)
  (when (< coffee-in-blood 10)
    (message "I am in dire need of more coffee!")))
my-awesome-function
ELISP> (my-awesome-function 11)
nil
ELISP> (my-awesome-function 7)
"I am in dire need of more coffee!"
ELISP>
ELISP> (+ 5 9)
14 (#o16, #xe, ?\C-n)
ELISP>
U:***- *ielm* All of 360 (13,7) (IELM:run on *ielm*) [99.9%]
```

Figure 1: \*Coffee scripting\* med Ielm.

## 7 Hooks

Indtil videre har ændringer til init-filen vi har kigget på galdt globalt, dvs hvis vi eksempelvis har skrevet (`highlight-numbers-mode 1`), så vil alle aktive buffere vise tal med en anden farve end bogstaverne - en meget nyttig indstilling til programmeringssprog, men måske ikke ønsket, når man skriver LaTeX. Her kommer hooks ind i billedet.

Et hook er en liste af funktioner, der skal kaldes på et bestemt tidspunkt. Der er fx et `suspend-hook`, som kaldes når Emacs lukker, og der er et hook for hvert major mode ("`M-x`" → "`c-mode`" aktiverer major mode for C programmering). For at aktivere markering af tal, men kun for programmeringssprog, kan følgende skrives i init-filen:

---

```
elisp - init.el
;; highlight numbers in programming modes
(add-hook 'prog-mode-hook 'highlight-numbers-mode)
```

---

Man kan også gøre flere ting på een gang ved at kalde en funktion, eller indskrive en lambda i stedet for (lambda = anonym funktion). Dette ikke meget mere komplicerede stykke kode kan også aktivere visning af kolonnenumre:

---

```
elisp - init.el
;; settings for all programming modes
(add-hook 'prog-mode-hook (lambda ()
  (highlight-numbers-mode)
  (column-number-mode)))
```

---

Læs mere om hooks i Emacs:

[https://www.gnu.org/software/emacs/manual/html\\_node/elisp/Hooks.html#Hooks](https://www.gnu.org/software/emacs/manual/html_node/elisp/Hooks.html#Hooks)

## 8 Automatisk backup

Som standard (ikke særlig tilfredsstillende) gemmer Emacs en masse backup-filer med tilde ~ i slutningen af filnavnet. Det ser generende ud, så det fikses 'nemt':

---

elisp - *init.el*

---

```
;; choose a directory for backup files
(defvar backup-directory "~/backups")
(if (not (file-exists-p backup-directory))
    (make-directory backup-directory t))

;; choose configuration for backup files
(setq
  make-backup-files t      ; backup a file the first time it is saved
  backup-directory-alist '((".*" . ,backup-directory)) ; save files in chosen
                          backup folder
  backup-by-copying t      ; copy the current file into backup directory
  version-control t        ; version numbers for backup files
  delete-old-versions t    ; delete unnecessary versions
  kept-old-versions 6      ; oldest versions to keep when a new numbered backup
                          is made (default: 2)
  kept-new-versions 9      ; newest versions to keep when a new numbered backup
                          is made (default: 2)
  auto-save-default t      ; auto-save every buffer that visits a file
  auto-save-timeout 40     ; number of seconds idle time before auto-save (
                          default: 30)
  auto-save-interval 400   ; number of keystrokes between auto-saves (default:
                          300)
)
```

---

Disse indstillinger kan selvfølgelig justeres efter behov, og er udelukkende indsat for demonstrations skyld.

## 9 Indentering

I en aktiv C buffer, tast "C-c C-o", og mini-bufferen vil vise den variabel, som definerer indenteringen på den pågældende linje. Tast "RET", og du har mulighed for at ændre indenteringen (eller tast "C-g" for at afbryde).

Det er værd at bemærke, at denne indentering er afhængig af, om Emacs er indstillet til at indentere med tabs eller spaces! Til CompSys anbefales det at indentere med 2 spaces, og dette opnås ved at tilføje følgende linjer til init-filen:

---

elisp - *init.el*

---

```
(setq indent-tabs-mode nil)
(setq-default tab-width 2)
```

---

### eksempel:

For at kunne indentere smertefrit med 2 spaces, kunne følgende linjer tilføjes til init-filen:

---

elisp - *init.el*

---

```
(defun my-c-mode-hook ()
  (setq-default c-basic-offset 2
                tab-width 2
                indent-tabs-mode nil)
  (c-set-offset 'defun-block-intro 2)
  (c-set-offset 'statement-block-intro 2)
  (c-set-offset 'comment-intro 0)
  (c-set-offset 'func-decl-cont 0)
)

(add-hook 'c-mode-hook 'my-c-mode-hook)
```

---

Hvis indentering 'driller', brug da metoden vist ovenfor: Undersøg, hvilken variabel, der definerer den pågældende linjes indentering, og tilføj rettelsen til init-filen.

## 10 Kompilering

Det kan være et tungt arbejde at kompilere en C fil, især hvis der skal linkes med flere biblioteker, defineres kompiler-version, optimeringsflag, output-fil, etc. Et hårrejsende eksempel på dette kunne være:

```
gcc -std=gnu11 -Wall -Werror -pedantic -o testfile testfile.c
```

Det ville være unødvendigt at skrive dette hver gang i terminalen (selvfølgelig findes der en Makefile!). En hurtig Emacs-løsning ville være følgende tilføjelse til init-filen:

elisp - *init.el*

---

```
(defun my-compile-function ()
  "Define custom compile settings relative to active 'major-mode'."
  (interactive)
  (defvar filename)
  (defvar comp-flags)
  (defvar output)

  (setq filename (file-name-nondirectory buffer-file-name))

  ;; c-mode
  (when (member major-mode '(c-mode))
    (setq comp-flags "-std=gnu11 -Wall -Werror -pedantic ")
    (setq output (concat "-o " (file-name-sans-extension filename) " "))
    (setq compile-command (concat "gcc " comp-flags output filename)))

  ;; ... other programming modes might be included here ...

  ;; COMPILE:
  (call-interactively 'compile)
  )

(global-set-key (kbd "<f5>") 'my-compile-function)
```

---

*"(interactive)" betyder i denne sammenhæng at funktionen vil optræde i menuen når man trykker "M-x".*

Det anbefales, at læseren afsætter lidt tid til at forstå, hvordan denne kommando fungerer. Prøv at køre den!

## 11 Ekstra

Der findes et utal af pakker til Emacs. Her vil blot blive nævnt et par af de mest betydningsfulde, når det gælder øget livskvalitet: (Husk at installere pakkerne via Emacs' pakke-manager først.)

### Syntaks-check

Det kan være en uvurderlig hjælp at få sin tekst-editor opsat med løbende syntaks-check af ens kode. Pakken 'flycheck' eksisterer til dette formål, og kræver ikke særlig meget at sætte op:

---

elisp - *init.el*

---

```
;; on the fly syntax check
(require 'flycheck)
(global-flycheck-mode)

;; package to show flycheck error after jumping to point (OPTIONAL)
(require 'flycheck-tip)
(flycheck-tip)
```

---

### Pop-up terminal

Pakken 'shell-pop' kan være smart hvis man hurtigt vil køre en terminalkommando inde fra den mappe, hvori den aktive buffer er hentet fra. Med et tryk på en knap (her er valgt "<f2>") popper der en terminal frem neden under den aktive buffer, og med et tryk på samme knap forsvinder den igen.

---

elisp - *init.el*

---

```
(require 'shell-pop)
(global-set-key (kbd "<f2>") 'shell-pop)
```

---

### Auto-completion

At få komplet auto-completion, completion-on-dot, header-file-indexing, og hvad det nu ellers hedder, kræver et større indgreb i ens Emacs-opsætning. Den nysgerrige læser kan kigge på denne tunge, men dybdegående guide: <http://tuhdo.github.io/c-ide.html#sec-2>. Nedenfor opridses et par forslag i denne retning.

#### auto-completion

Pakken 'company' leverer et framework til at komme med completion-forslag.  
(company = *complete anything*)

---

elisp - *init.el*

---

```
;; framework for auto-completion
(require 'company)
(global-company-mode 1)
```

---



---

### header-filer

Til 'company' kan der installeres en række udvidelser ("backends"), og en af dem er 'company-c-headers', som kan vise en liste af valg når man skriver `#include <...>` i toppen af et C-script. Følgende linjer kan da tilføjes til init-filen:

---

elisp - *init.el*

---

```
;; header completion c headers
(require 'company-c-headers)
(add-to-list 'company-backends 'company-c-headers))
```

---

---

### funktionsargumenter

Pakken 'function-args' kan vise signaturen for en funktion i auto-complete listen, som 'company' giver. Det er en ganske nyttig pakke, som også kan lave 'jump-to-definition', hvis den opsættes korrekt. (se mere: <https://github.com/abo-abo/function-args>)

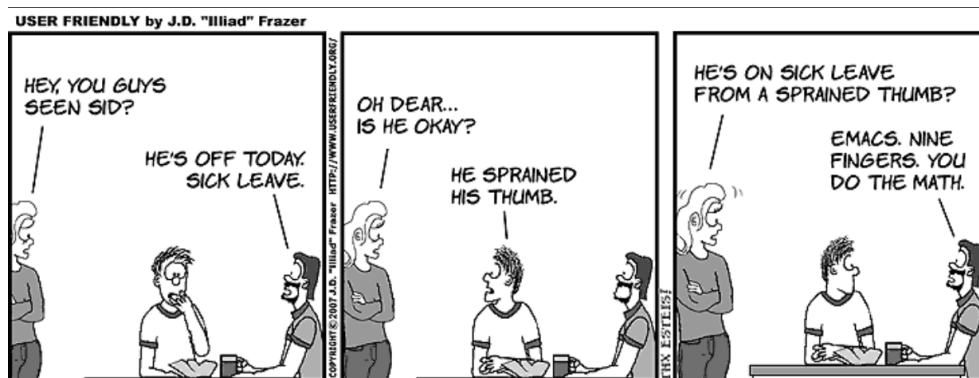
---

elisp - *init.el*

---

```
;; show function signatures
(require 'function-args)
(fa-config-default)
```

---



## Afrunding

Emacs er et utroligt versatilt værktøj. Det er kun en brøkdel af, hvad der findes af nyttige pakker og udvidelser, som der har været plads til i denne guide. Til trods for den høje alder, så lever Emacs i bedste velgående, og der udvikles stadig pakker. Det er mit håb, at denne guide har kunnet illustrere de centrale elementer i, hvordan man kan konfigurere og anvende Emacs. Min egen opsætning er tilgængelig på github: <https://github.com/alexpanter/emacs-repository>, og leveres udelukkende some reference, og uden garantier!