

---

## Objective Caml und XEmacs

---

Im Verlauf der Vorlesung wird ein sehr komplexes Programm entstehen. Um trotzdem den Überblick zu behalten und auch mit der ungewohnten OCaml-Umgebung gut zurechtzukommen, habe ich hier ein Paar Tipps zusammengeschrieben, die das Arbeiten mit OCaml extrem erleichtern.

1. **Benützt den XEmacs!** Alle Informationen und Downloads zum XEmacs gibt es auf <http://www.xemacs.org>.
2. Wie man den XEmacs zum Zusammenarbeiten mit OCaml kriegt, steht hier: <http://www-pu.informatik.uni-tuebingen.de/cc-0304/ocaml/xemacs.html>  
**Auf den Poolrechnern ist das bereits alles installiert.**
3. Wie man an den Code zur Vorlesung kommt, findet ihr hier: <http://www-pu.informatik.uni-tuebingen.de/cc-0304/code/>  
Denkt dran, den Code jedesmal nach dem auschecken mit `make` zu kompilieren. Falls es Probleme beim Kompilieren gibt, stellt sicher, dass es beim auschecken keine Konflikte gab (Konflikte in einer Datei werden in der CVS Ausgabe mit einem "C" (für Conflict) markiert. Konflikte können auftreten, wenn ihr Änderungen an Stellen gemacht habt, die im Repository ebenfalls verändert wurden. Manchmal hilft vor dem Kompilieren auch ein `make clean`.
4. Ich habe mir in meinem Verzeichnis `compilerbau/` eine Datei `cb.ml` erstellt, in der folgendes steht:

```
#use "load.ml";;

(* scanner example *)
Camllex.scan ['i'; 'f'; ' '; 'a'; '<'; 'b'; '\n'; 't'; 'h'; 'e'; 'n'; ' ';
              'a'; '\n'; 'e'; 'l'; 's'; 'e'; ' '; 'b'];;
Camllex.scan ['1'; '+'; '1'];;
```

In `load.ml` steht das:

```
#load "regexp.cmo"
#load "lexspec.cmo"
#load "lexstate.cmo"
#load "lex.cmo"
#load "camllex.cmo"
#load "camlsyn.cmo"
```

*Anmerkung:* `#use` benutzt man, um OCaml Sourcecode (Dateiendung `.ml`) einzubinden; `#load` um kompilierte OCaml Dateien (`.cmo`) zu laden.

Zudem habe ich noch die Datei `ocaml-cb` mit folgendem Inhalt erstellt und mit Ausführrechten versehen (`chmod a+x ocaml-cb`):

```
ocaml -I lex -I mini-caml
```

Diese drei Dateien benutze ich, um schnell die OCaml Umgebung, die man für das Implementieren und Testen des Vorlesungs-Compilers braucht, herzustellen:

- (a) Ich öffne im XEmacs die Datei `cb.ml`.
- (b) Mit `C-c C-s` starte ich den OCaml-Prozess im XEmacs.  
Bei der Frage welches Programm daraufhin gestartet werden soll, ersetze ich die Vorgabe `ocaml` mit `~/compilerbau/ocaml-cb`. Jetzt startet OCaml in einem neuen Buffer (namens `inferior-caml`) und kennt die Verzeichnisse, in denen es nach Dateien suchen muss.
- (c) Nun kann ich im `cb.ml`-Buffer mit dem Cursor auf die erste Anweisung gehen (`#load load.ml;;`) und mit `C-M-x` diese Anweisung in den OCaml-Buffer schicken. Alle in `load.ml` aufgelisteten Dateien werden geladen und können nun verwendet werden.

Ich habe in der Beispieldatei bereits zwei Testfälle für die erste Programmieraufgabe am Compiler aufgeführt. Zum ausführen hin mit dem Cursor und `C-M-x` drücken (natürlich müsst ihr die Lösung erst programmieren ;-)).

Habt ihr am Code irgendwas geändert, könnt ihr diesen Teil ebenfalls mit `C-M-x` in den OCaml-Prozess schicken. Eventuelle Fehlermeldungen werden dann dort angezeigt.

*Noch ein Tipp:* Mit `C-Tab` bzw. `C-x o` könnt ihr zwischen verschiedenen Buffern wechseln, zur OCaml-REPL gehts direkt mit `C-c C-s`.

Im OCaml-Buffer könnt ihr natürlich auch Befehle tippen. Mit `M-p` und `M-n` könnt ihr früher eingegebene Zeilen zurückholen.