C	Clight	CFML
`int *p = malloc(1);`	`(* declas: Etempvar tmp (void *\) Evar p (int *\) *) Sseq (Sbuiltin (Some p) malloc [1]) (Sassign (Evar p) (Etempvar tmp))`	`trm_let p heap (val_alloc 1)`
`*p = v;`	`Sassign (Ederef (Evar p)) v`	`(val_set p v)`
`*p;`	`Ederef (Evar p)`	`(val_get p)`
`p;`	`Evar p`	`p`
`free p;`	`Sbuiltin None free (Evar p)`	`(val_free p)`
`int x;`	En vrai difficile à voir comme cas `(* decla: Evar x (int) *)`	`trm_let x stack (val_alloc 1)`
`x = v;`	`Sassign (Evar x) v`	`(val_set x v)`
`x;`	`Evar x`	`(val_get x)`
`& x;`	`Eaddrof (Evar x)`	`x`
`register int x = 3;`	`Sset x (Eint 3)`	`trm_let x const (val_int 3)`
`x;`	`Etempvar x`	`(val_get x)`
`f(x);` (si résultat utile)	`(* decla: Etempvar tmp (void *\) *) Scall Some tmp f [x]`	`trm_let y const f [x]`
`f(x);` (sinon)	`Scall None f [x]`	`trm_let bind_anon const f [x]`
`if (e) then s1 else s2`	`Sifthenelse e s1 s2`	`(trm_ite e s1 s2)`
`while (e) s` (voir comment on compile si la cond a des side-effects (same pour ite))	`Swhile e s`	`trm_while e s`

Compilation phases:

`while t1 do t2`

```
while (1) {
    ⟦t1⟧_b;
    if (!b) {
      break;
    }
    t2;
}
```

Tout appel de fonction ->

`let y = f x in`

(entraîne la décla de y comme tempvar)