HCS Bootcamp 2: Emacs

Kenny Yu

How to turn this...

synch.h - emacs@appliance.localdomain

```
 */

#include <spinlock.h>

/*
 * Dijkstra-style semaphore.
 *
 * The name field is for easier debugging. A copy of the name is made
 * internally.
 */
struct semaphore {
        char *sem_name;
        struct wchan *sem_wchan;
        struct spinlock sem_lock;
        volatile int sem_count;
};

struct semaphore *sem_create(const char *name, int initial_count);
void sem_destroy(struct semaphore *);

/*
 * Operations (both atomic):
 *     P (proberen): decrement count. If the count is 0, block until
 *                   the count is 1 again before decrementing.
 *     V (verhogen): increment count.
 */
void P(struct semaphore *);
```

-:--- synch.h 37% L61 Git-master (C/l Abbrev)

Mark set

... into this.

```
[-] kern
  [-] arch
    [-] mips
      [+] conf
      [-] include
        [+] kern
        current.h
        elf.h
        specialre$
        spinlock.$
        thread.h
        tlb.h
        trapframe$
        types.h
        vm.h
      [+] locore
      [+] syscall
      [+] thread
      [+] vm
  [+] sys161
[+] compile
[+] conf
[+] dev
[+] fs
[-] include
  [+] kern
  addrspace.h
  array.h
  bitmap.h
  cdefs.h
  clock.h
  copyinout.h
  cpu.h
  current.h
  device.h
  elf.h
  emufs.h
  endian.h
  fs.h
  lib.h
  limits.h
  mainbus.h
  setjmp.h
  sfs.h
  signal.h
  spinlock.h
  spl.h
  stat.h
  stdarg.h
  synch.h
  syscall.h
  test.h
  thread.h
  threadlist.h
W-0 ...yyu-os161/kern/include
```

```
[-] = =synch.h
30 #ifndef _SYNCH_H_
31 #define _SYNCH_H_
32
33 /*
34  * Header file for synchronization primitives.
35  */
36
37 #include <spinlock.h>
38
39 /*
40  * Dijkstra-style semaphore.
41  *
42  * The name field is for easier debugging. A copy of the name is made
43  * internally.
44  */
45 struct semaphore {
46         char *sem_name;
47         struct wchan *sem_wchan;
48         struct spinlock sem_lock;
49         volatile int sem_count;
50 };
51
52 struct semaphore *sem_create(const char *name, int initial_count);
53 void sem_destroy(struct semaphore *);
54
55 /*
56  * Operations (both atomic):
57  *     P (proberen): decrement count. If the count is 0, block until
58  *                   the count is 1 again before decrementing.
59  *     V (verhogen): increment count.
60  */
61 void P(struct semaphore *);
62 void V(struct semaphore *);
63
64 /*
-uu-:---F1  synch.h        35% (45,18)    (C/l 80+ Abbrev)----4:33AM 1.16-----------------------------
[-] = =*eshell*
1 Welcome to the Emacs shell
2
3 /Users/kennyyu/Desktop/cs161/kennyyu-os161/kern/include $ ls
4 addrspace.h  copyinout.h  emufs.h   limits.h   spinlock.h  syscall.h        types.h    vnode.h
5 array.h      cpu.h        endian.h  mainbus.h  spl.h       test.h           uio.h      wchan.h
6 bitmap.h     current.h    fs.h      setjmp.h   stat.h      thread.h         version.h
7 cdefs.h      device.h     kern      sfs.h      stdarg.h    threadlist.h     vfs.h
8 clock.h      elf.h        lib.h     signal.h   synch.h     threadprivate.h  vm.h
9 /Users/kennyyu/Desktop/cs161/kennyyu-os161/kern/include $
-uuu:---F1  *eshell*       All (9,58)     (EShell)----4:33AM 1.16-----------------------------------
```

```
[+] Includes
[-] semaphore : struct
 `- [+] Attributes
[-] lock : struct
 `- [+] Attributes
[-] cv : struct
 `- [+] Attributes
[+] Variables
P (struct semaphore*) : void
V (struct semaphore*) : void
cv_broadcast (cv : struct cv*,lock : str$
cv_create (name : char*) : struct cv*
cv_destroy (struct cv*) : void
cv_signal (cv : struct cv*,lock : struct$
cv_wait (cv : struct cv*,lock : struct l$
lock_acquire (struct lock*) : void
lock_create (name : char*) : struct lock$
lock_destroy (struct lock*) : void
lock_do_i_hold (struct lock*) : bool
lock_release (struct lock*) : void
sem_create (name : char*,initial_count :$
sem_destroy (struct semaphore*) : void
W-3 synch.h
```

Getting Emacs

- On a mac:
  - `Probably already installed`
  - `Also aquamacs, a gui frontend for emacs`
- On ubuntu
  - `$ sudo apt-get install emacs`
- On fedora (e.g. CS50 Appliance)
  - `$ sudo yum install emacs`

Running Emacs

At the command line:

```
$ emacs [file-names-optional]
```

```
$ emacs hello.c world.c
```

Emacs Commands

- Most commands involve using the ctrl key (C-...) and the meta key (M-...) which is usually the esc key and/or option/alt key
- Basic commands
  - C-x C-s (save)
  - C-x C-f, then enter file name (open file)
  - C-x C-c (quit)

Movement

- Arrow keys for movement
  - C-f (forward char), C-b (backward char)
  - C-n (next line), C-p (previous line)
  - Use M-{fbnp} to jump words/blocks at a time
- More:
  - C-v (page down), M-v (page up)
  - C-a (beginning of line), C-e (end of line) [for macs, works in most other programs too!]
  - M-. (end of file), M-, (beginning of file)
  - C-l (recenter screen here, note that's the letter L)

Searching

- C-s enter word (forward search for text)
- C-r enter word (backward search for text)
- M-% (find and replace)

Editing Text

- More operations
  - C-d (backwards delete a char)
  - C-k (kill rest of line)
  - C-y (yank: paste what you just killed)
  - C-/ (undo)
- C-space (set mark point), C-g (quit current command)
  - Moving the cursor now allows you to highlight a region and perform operations on the region
  - C-w (delete region)
  - M-1 M-| [shell command on region]
    - Example: M-1 M-| sort

Window Management

- Windows are the regions, buffers are things you've opened that can go into regions
- Window management:
  - C-x 2 (split top-bottom)
  - C-x 3 (split left-right)
  - C-x o (switch windows)
  - C-x 0 (close current window)
  - C-x 1 (close all but this window)
- Buffer management
  - C-x C-b (see list of buffers open)
  - C-x b buffername (open buffer)
  - C-x k (kill current buffer)

Emacs Commands

- All these shortcuts are really just aliases for functions that were already pre-defined
- To execute any possible emacs command:
  - M-x command-name
- Examples:
  - M-x forward-char (usually C-f)
  - M-x eshell (start a shell in this current window)

Just the tip of the iceberg...

- Many more built-in commands
  - http://www.cs.rutgers.edu/LCSR-Computing/some-docs/emacs-chart.html
- No good text editor would be good without customization! ☺

Customizing Emacs

- Customize emacs with a language called elisp – language very similar to Scheme (functional programmers really like emacs!)
- 2 ways
  - 1) in your home directory, there is a .emacs file that gets run every time you start emacs
  - 2) or, in your home directory you have a .emacs.d that contains a file called init.el that gets run every time you start emacs
    - Better to use this way; can store other emacs-related things all together

Customizing Emacs

```
;; insert spaces for tabs
(setq-default indent-tabs-mode nil)

;; for emacs 23+, show line numbers on left
(global-linum-mode 1)

;; map C-o to ‘other window (same as C-x o)
(global-set-key (kbd “C-o”) ‘other_window)

;; you can load other emacs macros other people have written too!
;; you have to make sure that those files are currently in your
;; current emacs load-path
;; example: Put all files in my ~/.emacs.d onto my load path
(add-to-list ‘load-path “~/.emacs.d”)

;; load the definitions from a file, e.g. mouse control
(load-file “~/.emacs.d/mwheel.el”)
(require ‘mouse)
```

Lots of goodies online!

- emacs-goodies-el
  - http://packages.debian.org/sid/emacs-goodies-el
- My-macros
  - https://github.com/hcs/bootcamp-editors
  - Beware: non-standard key-bindings, only really tested on emacs22 on a mac, but should work for most environments!

My macros

- Show source tree
- Tabs for each buffer
- Code outline (show functions, classes, variable breakdown)
- Highlight over 80+ chars
- Mouse control
- Ocaml Tuareg Mode
- Solarized Color Theme
  - http://ethanschoonover.com/solarized
- Hotkey movement similar to Vim: M-{hjkl} and M-S-{hjkl} (shift)

Demo