Solução para o Mac OS X / Linux

diff --git a/deps/uv/src/unix/fsevents.c b/deps/uv/src/unix/fsevents.c

index 8143f7c..1259754 100644

--- a/deps/uv/src/unix/fsevents.c

+++ b/deps/uv/src/unix/fsevents.c

**@@ -76,6 +76,7 @@** typedef struct uv\_\_cf\_loop\_state\_s uv\_\_cf\_loop\_state\_t;

struct uv\_\_cf\_loop\_signal\_s {

QUEUE member;

uv\_fs\_event\_t\* handle;

+ unsigned int closing:1;

};

struct uv\_\_fsevents\_event\_s {

**@@ -98,7 +99,9 @@** struct uv\_\_cf\_loop\_state\_s {

/\* Forward declarations \*/

static void uv\_\_cf\_loop\_cb(void\* arg);

static void\* uv\_\_cf\_loop\_runner(void\* arg);

-static int uv\_\_cf\_loop\_signal(uv\_loop\_t\* loop, uv\_fs\_event\_t\* handle);

+static int uv\_\_cf\_loop\_signal(uv\_loop\_t\* loop,

+ uv\_fs\_event\_t\* handle,

+ unsigned int closing);

/\* Lazy-loaded by uv\_\_fsevents\_global\_init(). \*/

static CFArrayRef (\*pCFArrayCreate)(CFAllocatorRef,

**@@ -387,7 +390,8 @@** static void uv\_\_fsevents\_destroy\_stream(uv\_loop\_t\* loop) {

/\* Runs in CF thread, when there're new fsevent handles to add to stream \*/

-static void uv\_\_fsevents\_reschedule(uv\_fs\_event\_t\* handle) {

+static void uv\_\_fsevents\_reschedule(uv\_fs\_event\_t\* handle,

+ unsigned int closing) {

uv\_\_cf\_loop\_state\_t\* state;

QUEUE\* q;

uv\_fs\_event\_t\* curr;

**@@ -486,7 +490,7 @@** final:

\*

\* NOTE: This is coupled with `uv\_sem\_wait()` in `uv\_\_fsevents\_close`

\*/

- if (!uv\_\_is\_active(handle))

+ if (closing)

uv\_sem\_post(&state->fsevent\_sem);

}

**@@ -676,7 +680,7 @@** void uv\_\_fsevents\_loop\_delete(uv\_loop\_t\* loop) {

if (loop->cf\_state == NULL)

return;

- if (uv\_\_cf\_loop\_signal(loop, NULL) != 0)

+ if (uv\_\_cf\_loop\_signal(loop, NULL, 0) != 0)

abort();

uv\_thread\_join(&loop->cf\_thread);

**@@ -753,7 +757,7 @@** static void uv\_\_cf\_loop\_cb(void\* arg) {

if (s->handle == NULL)

pCFRunLoopStop(state->loop);

else

- uv\_\_fsevents\_reschedule(s->handle);

+ uv\_\_fsevents\_reschedule(s->handle, s->closing);

QUEUE\_REMOVE(item);

uv\_\_free(s);

**@@ -762,7 +766,9 @@** static void uv\_\_cf\_loop\_cb(void\* arg) {

/\* Runs in UV loop to notify CF thread \*/

-int uv\_\_cf\_loop\_signal(uv\_loop\_t\* loop, uv\_fs\_event\_t\* handle) {

+int uv\_\_cf\_loop\_signal(uv\_loop\_t\* loop,

+ uv\_fs\_event\_t\* handle,

+ unsigned int closing) {

uv\_\_cf\_loop\_signal\_t\* item;

uv\_\_cf\_loop\_state\_t\* state;

**@@ -771,6 +777,7 @@** int uv\_\_cf\_loop\_signal(uv\_loop\_t\* loop, uv\_fs\_event\_t\* handle) {

return -ENOMEM;

item->handle = handle;

+ item->closing = closing;

uv\_mutex\_lock(&loop->cf\_mutex);

QUEUE\_INSERT\_TAIL(&loop->cf\_signals, &item->member);

**@@ -833,7 +840,7 @@** int uv\_\_fsevents\_init(uv\_fs\_event\_t\* handle) {

/\* Reschedule FSEventStream \*/

assert(handle != NULL);

- err = uv\_\_cf\_loop\_signal(handle->loop, handle);

+ err = uv\_\_cf\_loop\_signal(handle->loop, handle, 0);

if (err)

goto fail\_loop\_signal;

**@@ -873,7 +880,7 @@** int uv\_\_fsevents\_close(uv\_fs\_event\_t\* handle) {

/\* Reschedule FSEventStream \*/

assert(handle != NULL);

- err = uv\_\_cf\_loop\_signal(handle->loop, handle);

+ err = uv\_\_cf\_loop\_signal(handle->loop, handle, 1);

if (err)

return -err;