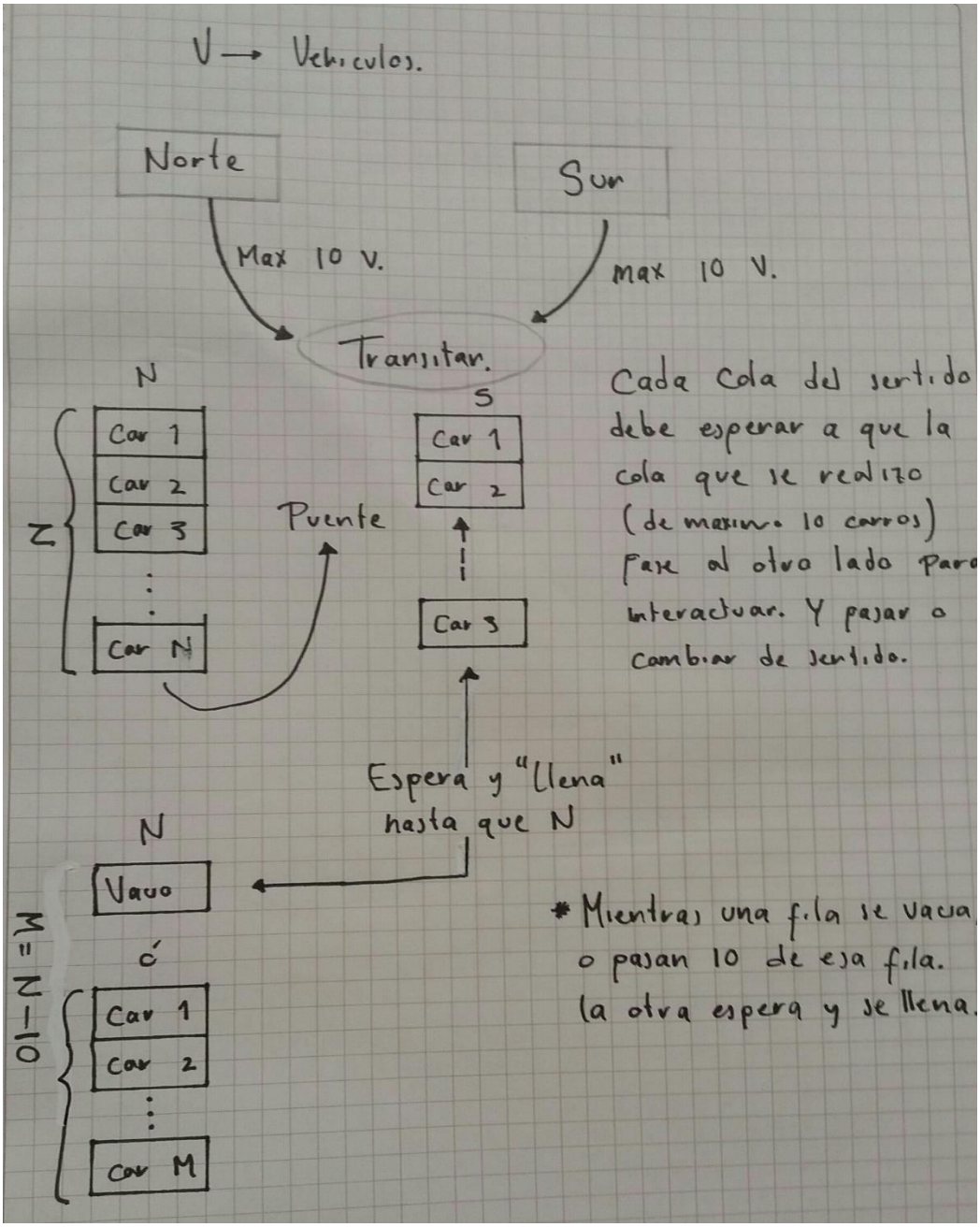
Santiago de Cali, 4 de Abril de 2016
Universidad Icesi
Sistemas Operativos
Jaime Vélez E. - 12207008
URL: https://github.com/jaimevlz7/icesi_examen_1
Examen Parcial 1:

SINCRONIZACION DE PROCESOS:

Para empezar; el puente consta de una capacidad total de un máximo de 10 carros sin importar su peso. Permite un único sentido (tránsito de carros y dirección) por turno. Para el desarrollo tomare como sentidos Norte y Sur (Izquierda o derecha también).



Se realizara un programa en java (Eclipse) para resolver el problema anterior, que contenga 4 clases **Norte, Sur, Puente** y **Ejecutable**.

Las clases **Norte** y **Sur<<Izquierda o Derecha>>** implementan los hilos para desarrollar la actualización de cuantos carros llegan. La clase consta de la siguiente declaración de atributos:

private Puente **puente**;
private Semaphore **semaforo**;

Haciendo uso (ambas clases) del semáforo para respetar el trámite (método) de “transitar” y que algún otro hilo interfiera con el correcto funcionamiento del método y su salida (Para ambas clases igual, únicamente cambiando ciertos atributos de **Norte** o **Sur<<Izquierda o Derecha>>**, respectivamente).

```
private Puente puente;
private Semaphore semaforo;

public Norte(Puente p){
    puente = p;
}
public void run() {

    while (true) {

        int tiempo = 1 + (int) (Math.random() * 10);
        System.out.println("Llegara un carro en:" + tiempo + " segundos");
        puente.setCantN(1);
        try {
            sleep(tiempo * 1000);
        } catch (InterruptedException e) {

        }
        try {
            semaforo.acquire();
            puente.transitar();
            semaforo.release();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }



    }
}
```

La clase **Puente** la encargada de administrar que lado del puente es el que tiene “la vía” y hacer todas las validaciones correspondientes, como no dejar pasar más de 10 carros y no permitir que transiten en el mismo sentido los autos, por lo cual también posee los atributos de

cantidad máxima permitida y saber cuántos autos hay a cada lado, además de 1 variable que permite saber desde que lado están transitando los vehículos para hacer ciertas verificaciones, implementa también el semáforo que permite la no simultaneidad de procesos pero si la sincronización de estos mismos. Tiene la siguiente declaración de atributos:

public final static int **CAP_MAX** = 10;
private boolean **norte**;
private boolean **sur**;
private int **cantS**;
private int **cantN**;
private Semaphore **semaforo**;

Esta clase implementa los métodos utilizados para el tránsito que son: transitar(), transitarSur(), transitarNorte();

```
public void transitar(){
    if(norte ==false && sur == false){
        norte = true;
        transitarNorte();
        System.out.println("Esta transitando norte");
    }
    if(norte==true){
        norte = false;
        sur = true;
        transitarSur();
        System.out.println("Esta transitando sur");
    }else{
        sur = false;
        norte = true;
        transitarNorte();
        System.out.println("Esta transitando norte");
    }
}
```

```
    private void transitarSur() {
        if(cantS <= CAP_MAX){
            cantS = 0;
            System.out.println("Cantidad de carros en sur:" + cantS);

        }else{
            cantS-=10;
            System.out.println("Cantidad de carros en sur:" + cantS);
        }
    }
    private void transitarNorte() {
        if(cantN <= CAP_MAX){
            cantN = 0;
            System.out.println("Cantidad de carros en norte:" + cantN);

        }else{
            cantN-=10;
            System.out.println("Cantidad de carros en norte:" + cantN);

        }
    }
```

La clase **Ejecutable** contiene el main del programa y permite arrancar e inicializar las clases que vamos a usar además de inicializar los hilos de cada lado:

```
public class Ejecutable {

    public static void main(String main[]){

        Puente p = new Puente();
        Norte n = new Norte(p);
        Sur s = new Sur(p);

        n.start();
        s.start();

    }

}
```

Ejemplo de sincronización:

- En una línea de montaje existe un robot programable encargado de colocar 3 productos diferentes (1, 2 o 3) sacados o “elaborados” aleatoriamente. Otros 3 robots, retiran los productos de la cadena de montaje para empacar, teniendo en cuenta que cada robot solo puede coger un producto de su tipo (tipo programado) ignorando los que no son de su tipo. Se quiere llevar un control del total de productos empaquetados sin importar su tipo. La cadena de montaje solo soporta tener 10 productos al tiempo. Si hay algún

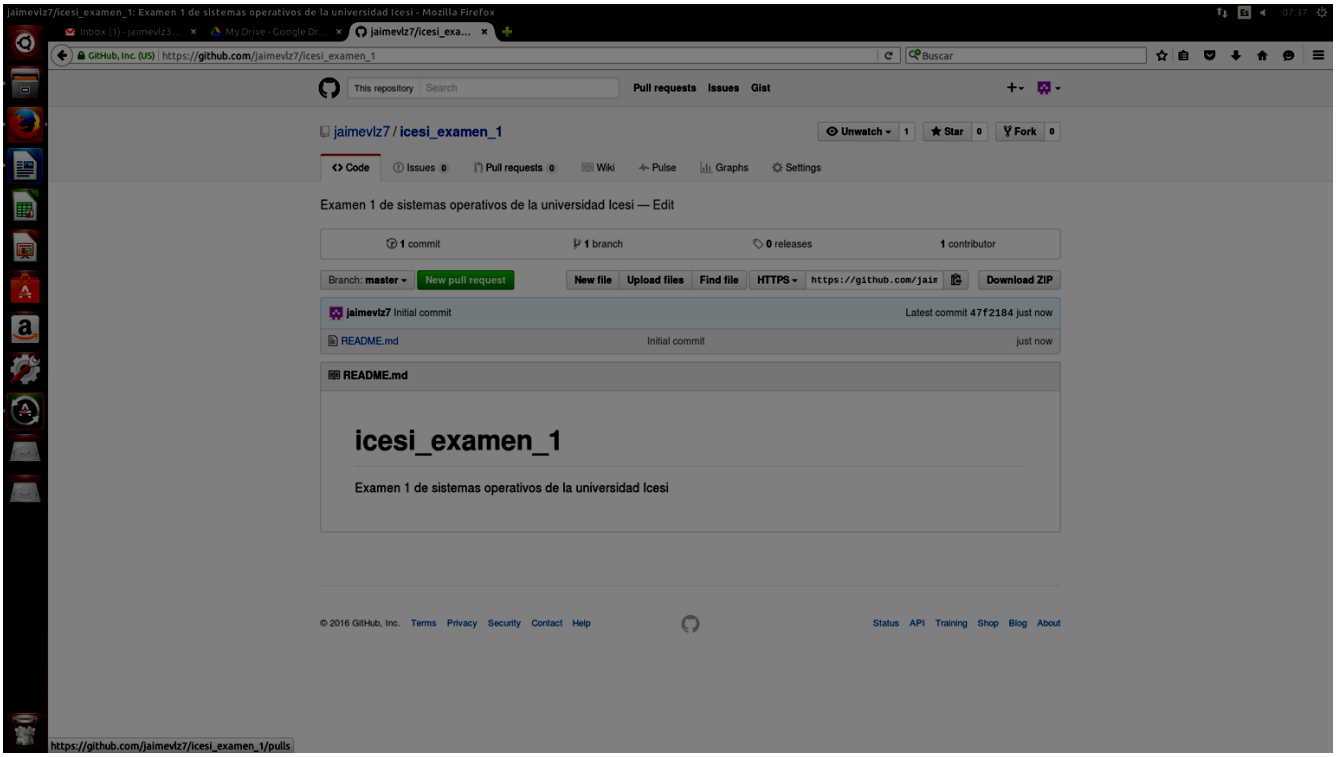
producto de su tipo lo retiran de la cadena (sólo 1 producto cada vez) y la posición queda libre para colocar nuevos productos, en caso contrario se quedan a la espera de que haya nuevos productos. Los robots empaquetadores de distinto tipo funcionan a la vez. Tanto el colocador como los empaquetadores nunca acaban. Cada vez que un robot empaquetador procesa un producto, la cuenta total de productos empaquetados debe aumentar.

La solución para el anterior punto permite la sincronización debido a que el sistema de reconocimiento de los empacadores debe esperar y analizar los recursos para poder tomar un producto de cierto tipo y tener en cuenta las actualizaciones automáticas cuando se empaca un producto o se coloca en la línea de montaje. Además de controlar también el tráfico y la puesta o no de productos de x tipo por el robot colocador. Ya que en algún momento la maquina puede haber actualizado mal en alguna empaquetada y hacer que el robot colocador deposite otro producto cuando en realidad no puede. O que alguna de las maquinas empacadoras quieran sacar algo cuando de verdad no pueden o no hay ese recurso,

Este problema se puede resolver haciendo uso de semáforos para sincronizar los procesos cuando se actualicen los productos empacados, cuando se pone un producto de x tipo y cuando se sacan para permitir un correcto funcionamiento.

GIT

Para la parte del uso de git y github, creo un nuevo repositorio para subir el parcial y todos los archivos asociados a este, llamado icesi_examen_1 como se muestra a continuación (Creando también el documento README.md):

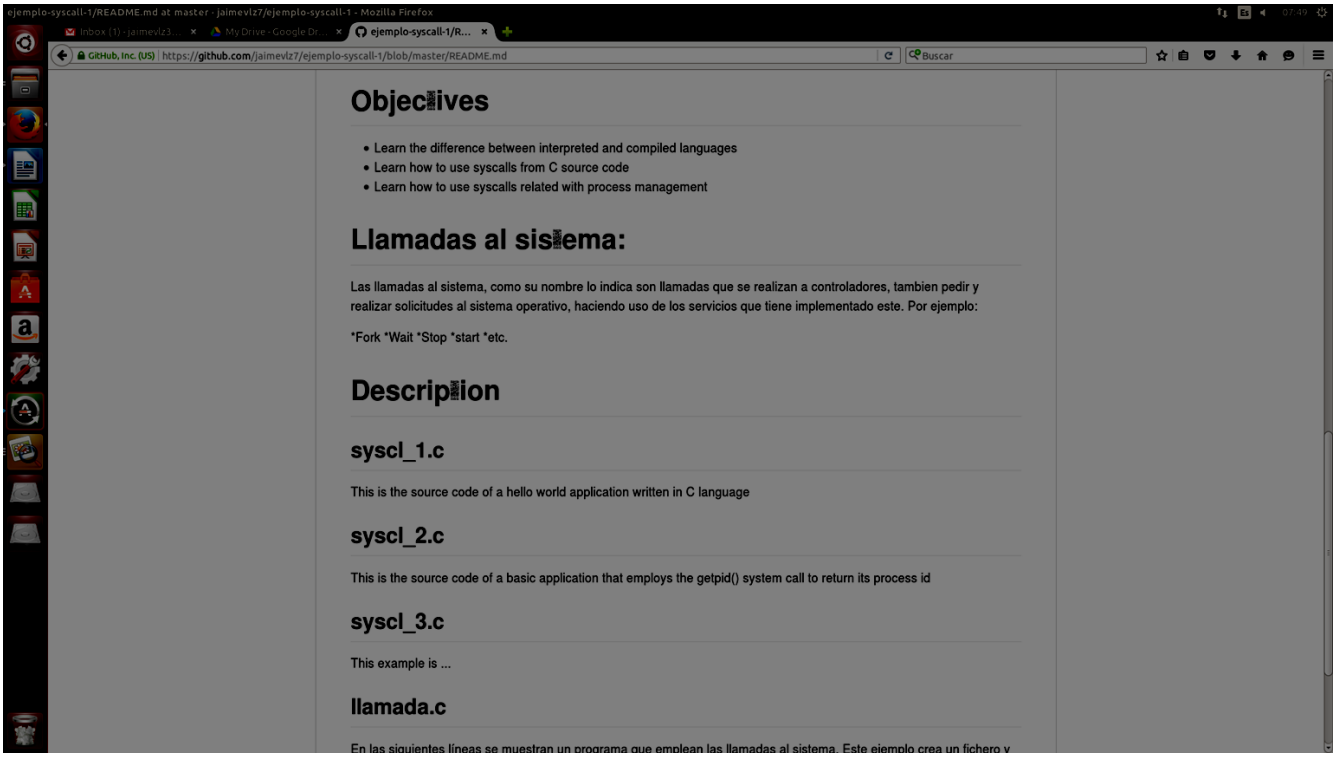


FORK:

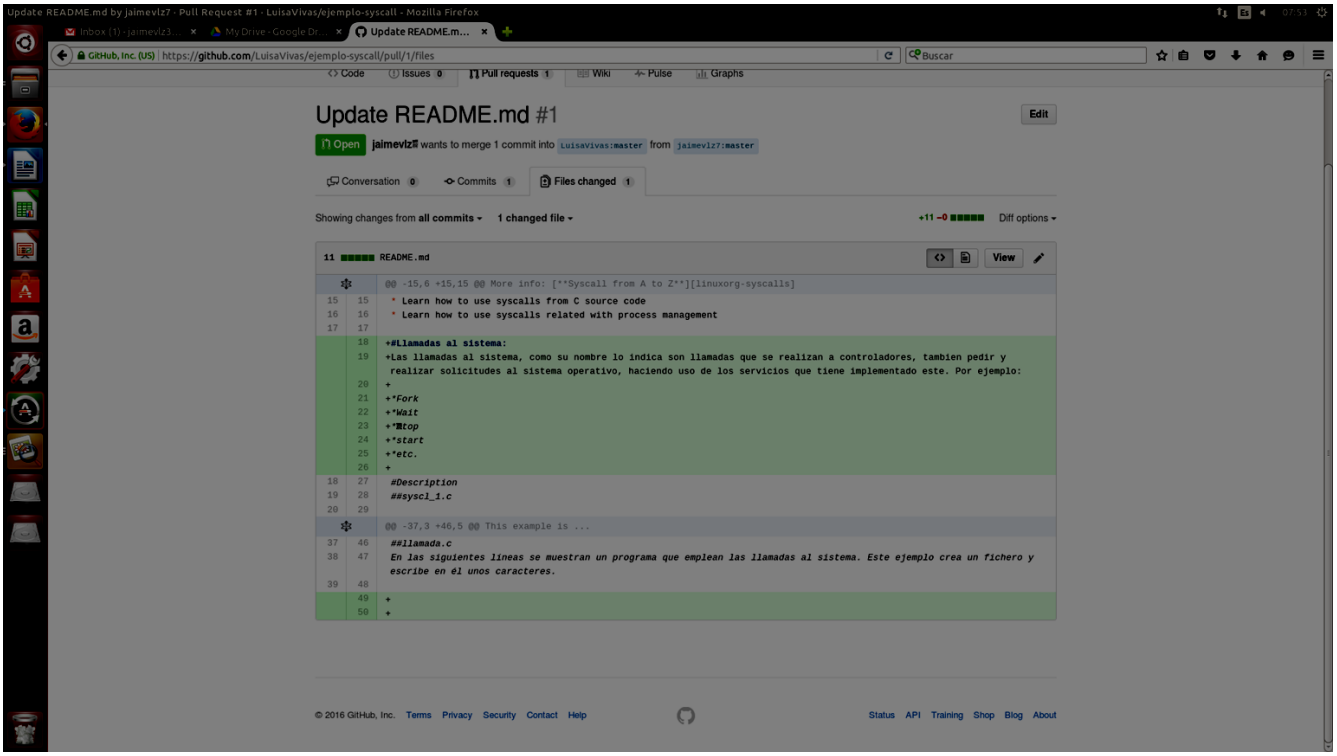
Realizo un fork a Luisa (Nombre de usuario: LuisaVivas) al repositorio ejemplo-syscall-1 y modifico el README.md añadiendo la definición básica y teórica de las llamadas al sistema así:
#Llamadas al sistema:

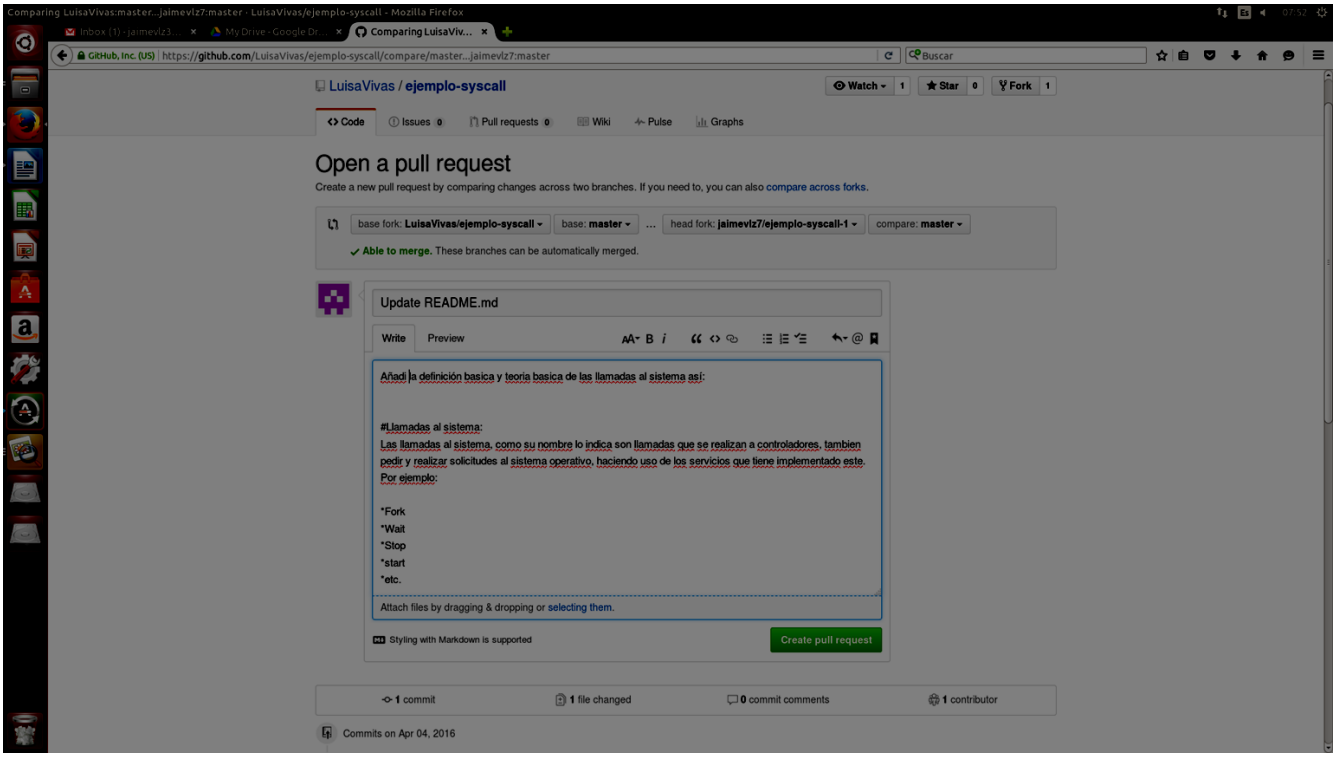
Las llamadas al sistema, como su nombre lo indica son llamadas que se realizan a controladores, también pedir y realizar solicitudes al sistema operativo, haciendo uso de los servicios que tiene implementado este. Por ejemplo:

*Fork
*Wait
*Stop
*start
*etc.



Después, como dice el enunciado hago el pull request y obtengo los cambios que realice a su documento y agrego comentario pertinente sobre los cambios que realice en el documento de README.md de LuisaVivas.




Comparing LuisaVivas:master...jaimevlz7:master · LuisaVivas/ejemplo-syscall - Mozilla Firefox
Comparing LuisaViv...
https://github.com/LuisaVivas/ejemplo-syscall/compare/master...jaimevlz7:master
LuisaVivas / ejemplo-syscall
Watch 1
Star 0
Fork 1
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Open a pull request
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.
base fork: LuisaVivas/ejemplo-syscall
base: master
head fork: jaimevlz7/ejemplo-syscall-1
compare: master
Able to merge. These branches can be automatically merged.
Update README.md
Write
Preview
Añadi la definición basica y teoria basica de las llamadas al sistema así:
#Llamadas al sistema:
Las llamadas al sistema, como su nombre lo indica son llamadas que se realizan a controladores, tambien pedir y realizar solicitudes al sistema operativo, haciendo uso de los servicios que tiene implementado este.
Por ejemplo:
*Fork
*Wait
*Stop
*start
*etc.
Attach files by dragging & dropping or selecting them.
Styling with Markdown is supported
Create pull request
1 commit
1 file changed
0 commit comments
1 contributor
Commits on Apr 04, 2016