**Sistemas Operativos-lab1:**

tar tf: p ver un archivo tar

~/Downloados/kickstart.tar.gz : ~ es el home

tar xf p extraer todos los archivos ~/Downloados/kickstart.tar.gz

mv kickstart/sistop20g3

parser.o compilado p poder llamar la función solo necesitamos el encabezado de la función

ls -lr: para ver todos los contenidos

Para comenzar con un repositorio git comenzar con:

$git init -> cuando pones esto git convierte el repositorio actual en un repositorio git(convierte todo en un entorno donde sabe trabajar)

(si nos fijamos: ls -la-> git crea un repositorio oculto donde tiene todo lo que necesita p hacer su magia )

una vez que creamos el repositorio si hacemos:

$git status

te dice que nadie hizo ningun commit que nadie puso archivos en el repositorio

y que los archivos que estan en el repositorio no los esta rastreando asi que lo que hacemos es agregar todo al repo de git:

$git add\*

$ git status

entonces git me dice que todos estos archivos que estaban en el tar.gz los estoy por agregar en el commit al repositorio

$git commit

-> arriba a la izquierda siempre poner un mensaje descriptivo de lo que estoy haciendo

del tipo: codigo original de la catedra

$guardo //no vi el commando

me fijo:

$ git status

on branch master ( no hay nada nuevo para comitear entonces todos los cambios ya estan subidos)

$ git log

commit ...dhfjrhfj (HEAD->master)

-para mandar a bitbucket:

en la pagina dice:

## Get your local Git repository on Bitbucket

Step 1: Switch to your repository's directory

cd /path/to/your/repo

Step 2: Connect your existing repository to Bitbucket

git remote add origin https://IsabelRivadero@bitbucket.org/sistop-famaf/so20lab1g23.git

git push -u origin master

(con eso se sube todo a nuestro repositorio)

estos comandos ponerlos en la terminal

-git remote add origin https://IsabelRivadero@bitbucket.org/sistop-famaf/so20lab1g23.git

(que al repositorio se le agregue un repositorio remoto que se va a llamar origin, el repo remoto tiene algunas características especiales usa git por defecto. el repo remoto va a estar en esa direccion:<https://IsabelRivadero@bitbucket.org/sistop-famaf/so20lab1g23.git>)

-una vez que lo subimos hay que subirle a origin nuestra rama master: git push -u origin master

en este punto ya estaria en bickbucket estartia todo el codigo que baje del kickstart

te pide la contraseña de bitbucket para descargar,

para hacer commits es fácil, desde la carpeta del proyecto lo unico que tenés que hacer es:

-git status (te muestra en rojo los archivos modificados)

-git add \* (para añadir todos los archivos)

-git status (si te muestran los archivos modificados en verde es porque se añadieron correctamente)

-git commit -m "Mensaje significativo para saber que hiciste"

-git push

-Para el martes implementar el modulo command:

**TADs pipeline y scommand**

A partir de la lectura de man bash en su sección SHELL GRAMMAR, se extrae la gramática que maneja el *shell*. Esta se divide en 3 capas: comando simple, tubería y lista, en orden creciente de complejidad. Limitaremos la implementación a los 2 primeros niveles.

● Un **comando simple** (scommand) es una secuencia de palabras donde la primera es el comando y las siguientes sus argumentos. Resultan opcionales dos redirectores, uno de entrada y otro de salida. Ejemplos:

ls -l Makefile wc archivo.c > estadisticas.txt

-En la carpeta correspondiente:

$vim command.h (con kate kate command.h &)

abre un editor en la consola:

typedef struct scommand\_s \* scommand; //en ningun momento decimos que tipo tiene ese struct, sino que decimos que con que me pases un puntero yo se que hacer

todos esos detalles estan ocultos en el archivo .c( command.c) donde vamos a implementarlos

para empezar a trabajar, si el encabezado es vim command.h entonces el archivo donde vamos a comenzar a hacer va a ser

$vim command.c

-revisar que le hace falta a scommand.c

le hace falta guardad el comando y la lista de parametros y las dos redicreecciones: redireccion de entrada y redireccion de salida

* cuando incluyamos siempre poner los include “loqueusamos.h”
* #include modulos
* #include <librerias>

Makefiles:

test: $(objects) -> ingredientes que hacen falta para generar el archivo de salida

-para generar el command.o a traves del command.c que acabo de escribir:

$make test-command

*//STUB implementar*

*//el codigo de la funcion es nada*

**void** scommand\_push\_back(scommand self, **char** **\*** argument){

}

pregunta:

que es lo que tendría que devolver cuando elvalor de retorno sea algun tipo de punterno? tenemos que ponermnos a definir una estructura y definir la memoria y resolver ese puntero? o hay una forma mas facil?

funcion que devuelve un scommand o pipeline:

pipeline pipeline\_new(**void**); //un puntero que se puede devolver es un puntero NULL

si tuvieramos queimplementar un atd de una funcion que transforma un elemento en otro

scommand scommand\_destroy(scommand self);

en ese caso podriamos devolver el mismo self

**glib: (min48 del video)**

***--heap* (montículo) y la pila de llamadas (*stack*) --**

El montículo libre, zona libre, almacenamiento libre o *heap* es una estructura dinámica de datos utilizada para almacenar datos en ejecución. A diferencia de la pila de ejecución que solamente almacena las variables declaradas en los bloques previo a su ejecución, el *heap* permite reservar memoria dinámicamente, es decir, es el encargado de que la «magia» de la memoria dinámica ocurra. Las variables globales y estáticas también son almacenadas en él.