

Arquitectura y Sistemas Operativos

Tecnicatura Universitaria en Programación (TUP) – UTN FRBB

Profesor: Gustavo Ramoscelli

Ayudantes: Leandro M. Regolf - Sergio Antozzi

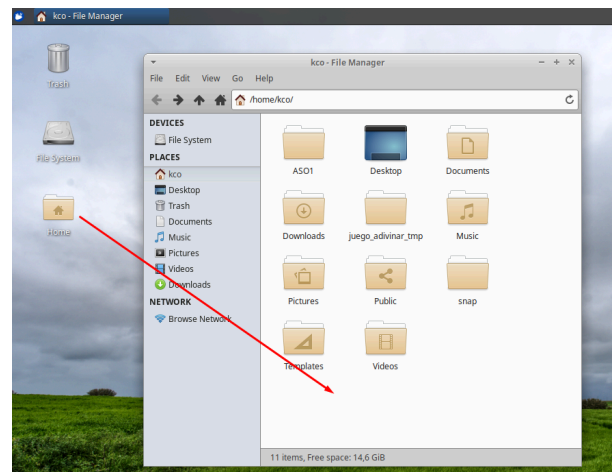
Trabajo Práctico Nº 2

Generalidades Linux / Git (control de versiones) / GCC (GNU Colección de Compiladores)

- 1- Generalidades y conceptos básicos de Linux
- 2- Algunos comandos BASH
- 3- Instalación de GCC (compiladores varios considerados oficiales de Linux)
- 4- Instalación de Git (control de versiones) y personalización
- 5- Conceptos básicos de Git
- 6- Creando el primer repositorio con Github
- 7- Compilado del programa procesos.c (código fuente en lenguaje C)
- 8- Ejecución de dicho programa
- 9- Listado de procesos corriendo
- 10- Captura de pantalla de la consola
- 11- Pusheando el TP2 a Github
- 12- Entrega del TP2

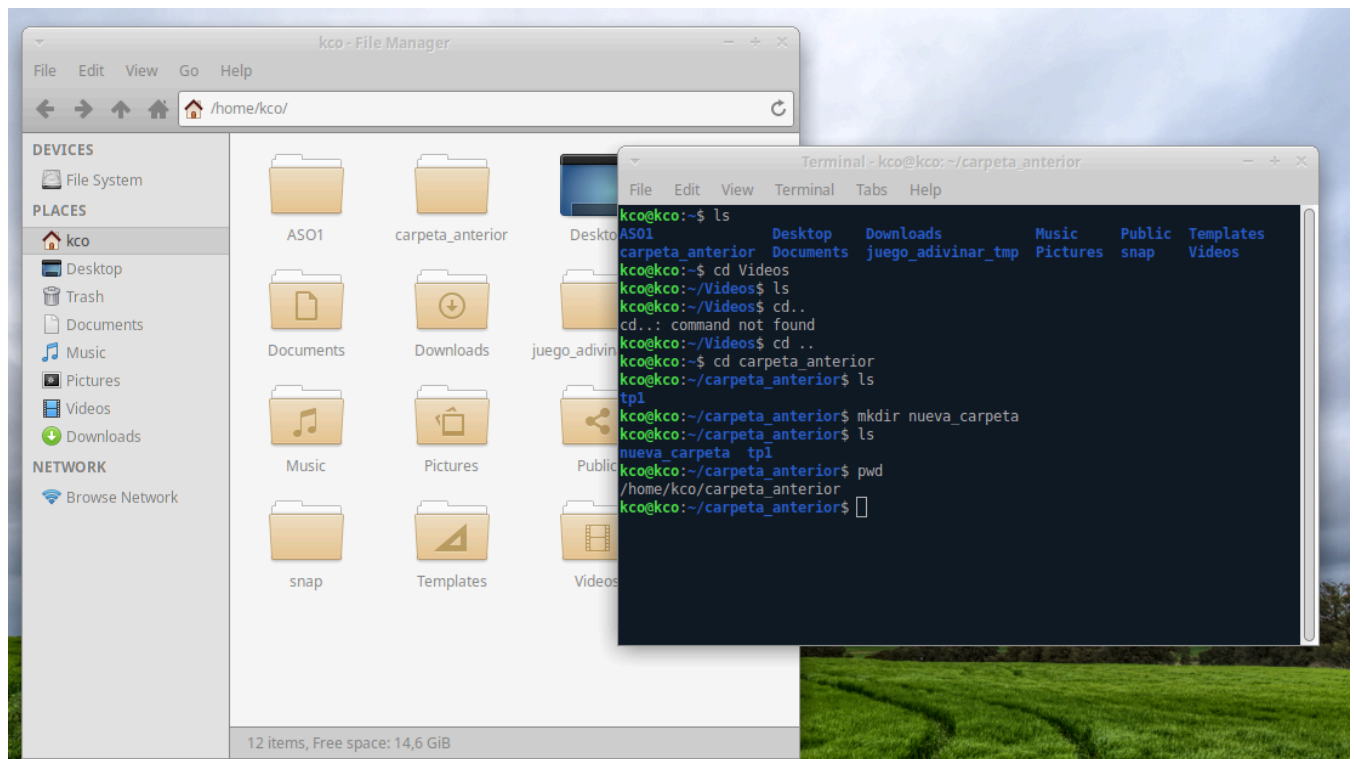
1- Conceptos básicos de Linux

- Linux es un **Sistema Operativo** de código abierto, creado por Linus Torvalds y mantenido en la actualidad por una amplia comunidad de desarrolladores
- Hace foco en la integridad y seguridad informática del sistema, por estos motivos es muy utilizado en servidores de todo tipo
- Al momento de la instalación se crea un usuario que cuenta con los permisos o **privilegios** más elevados para realizar cambios en el Sistema Operativo. Este usuario es conocido como “**root**”
- Los usuarios “normales”, por cuestiones de seguridad, poseen **permisos limitados** a tareas cotidianas, las tareas mas avanzadas de configuración se reservan al usuarios con mayores privilegios (como el usuario root)
- Dentro de **/home/<nombreusuario>** se encuentran todos los datos, archivos y contenidos específicos y privados de cada usuario del sistema



2- Algo de BASH e interfaz de usuario en línea de comandos

- La **terminal** nos permite interactuar con la computadora través del Sistema Operativo, es una interfaz del S.O.
 - El **BASH** es un Shell de la terminal, permite ejecutar comandos escritos
 - El BASH es **Case Sensitive** -> por lo que distingue o diferencia entre mayúsculas de minúsculas
 - Posicionados en la carpeta del usuario abrimos una terminal con el menú desplegable o bien con el atajo de teclado **Ctrl + Alt + T**
 - Comando **ls** -> permite listar los archivos dentro de la carpeta o directorio
 - Comando **cd** -> permite cambiar a dicha carpeta **cd <nombre carpeta>**
 - Comando **cd ..** -> sale de la carpeta actual, vuelve un paso hacia atrás en el árbol de carpetas
 - Comando **mkdir** -> crea nueva carpeta **mkdir <nombre nueva carpeta>**
 - Comando **pwd** -> permite ver la ruta actual
 - Comando **./** -> permite ejecutar un archivo binario ejecutable **./<nombre archivo ejecutable>**
 - Comando **sudo** -> permite elevar **temporalmente** los privilegios del usuario actual como si este fuera **root** o **super usuario** (siempre y cuando este usuario se encuentre en la lista de sudoers, cosa que sucede por defecto al momento de la instalación del S.O. en una instalación normal)
- (Las imágenes son de referencia y es muy posible que difieran en gran medida)



Nota:

El comando `./` y el comando **sudo** fueron necesario utilizarlos al instalar VSCode, por un lado indicamos con `./` que ejecute el paquete **.deb** que habíamos descargado, pero solo lo pudimos realizar con privilegios de **root** o **super usuario** por eso utilizamos

```
sudo apt install ./code_1.70.2-1660629410_amd64.deb
```

3- Instalando el Compilador GCC (GNU C Compiler)

- En Linux se utilizan gestores de paquetes para instalar software, existe también la posibilidad de instalar software adicional de manera gráfica, necesarios para administrar las dependencias de paquetes al instalar un determinado programa
- En Xubuntu viene configurado por defecto un gestor de paquetes llamado **APT**
- Este gestor nos permite instalar y desinstalar aplicaciones desde la línea de comandos de manera muy sencilla
- Para instalar el **GCC**:

```
sudo apt install gcc
```

4- Instalando Git

- Como primer paso se debe generar un usuario en la plataforma github <https://github.com/>, y loguearse en el navegador
- Luego, instalar **Git** en Linux:

```
sudo apt install git
```

- Luego de instalar **Git**, es necesario dirigirse a github.com en el explorador y generar un **Personal Token**, eventualmente lo necesitaremos al utilizar Git desde la línea de comandos.

Seguir esta guía:

<https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token>

- En el paso 8 de esa guía tildar todos los checkboxes, el **token** generado **guardarlo con aprecio** en algún archivo de texto o mail ya que desaparecerá y obligará a tener que crear otro. Este token se requerirá al momento de la autenticación.

5- Git básico (muy)

Cuando trabajamos con muy pocos archivos durante un proceso de creación o desarrollo, el control de versiones de archivos parece trivial. No es así cuando los cambios se realizan en varios archivos dentro de un mismo proyecto (a la vez) de manera relacionada, y cuando varios usuarios deben registrar y

replicar dichos cambios.

Git nos permite mantener y llevar un registro y control de cambios en nuestros archivos. Esto se conoce como **Control de versiones**.

Podemos mantener un control de versiones de manera local, pero Git también nos permite trabajar con **repositorios**, o **repos**, en **servidores externos**, permitiendo así compartir archivos y registros históricos incluso con usuarios en otros equipos.

Git detectará cambios en los archivos del directorio que le indiquemos, y cuando deseemos que dichos cambios queden “plasmadas” o registrados como una versión, se lo indicamos junto con un comentario descriptivo acorde.

Debemos configurarlo / personalizarlo:

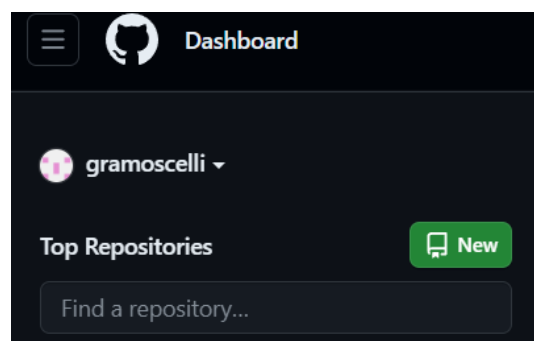
```
git config --global user.email <mail_usuariogit>  
git config --global user.name <nombre_usuariogit>
```

Una vez dentro del directorio que deseamos trackear o realizar un control de versión podemos utilizar distintos comandos (no es necesario utilizarlos ni ejecutarlos en este punto):

- **git init** -> inicializa un repositorio en la carpeta donde estemos parados
- **git add -A** -> indica que trackee (registre) todos los archivos con modificaciones realizadas en dicho directorio, pero aún no lo agrega a la “versión”
- **git commit -m "primer commit"** -> recién ahí guarda la versión trackeada con el comando anterior. Notar que el comentario al realizar un commit es obligatorio
- **git clone <nombre_repositorioaclone>** -> crea una copia local en la carpeta actual de un repositorio remoto indicado en nombre_repositorio
- **git push <origin main>** -> sube los cambios trackeados localmente al repositorio remoto

6- Creando el primer repositorio con Github

- Primeramente debemos loguearnos con nuestra cuenta personal en github.com, y permanecer logueados
- Creamos un repositorio público vacío a través de la web con nombre **ASO2024TPs**, la URL de dicho repo tendrá la forma como **[nombre_usuario]/ASO2024TPs** mediante el botón “NEW” que aparece arriba a la izquierda:



- Luego vamos a Linux para clonar localmente el repositorio remoto que creamos arriba mediante el comando:

```
git clone https://github.com/ [nombre_usuario]/ASO2024TPs
```

NOTA: Todos los comandos se introducen usando la consola o *shell* de Linux.

- Una vez clonado el repositorio, ingresamos dentro de la carpeta recién creada: ASO2024TPs
- Luego creamos una carpeta con nombre TP2 usando el comando:

```
mkdir TP2
```

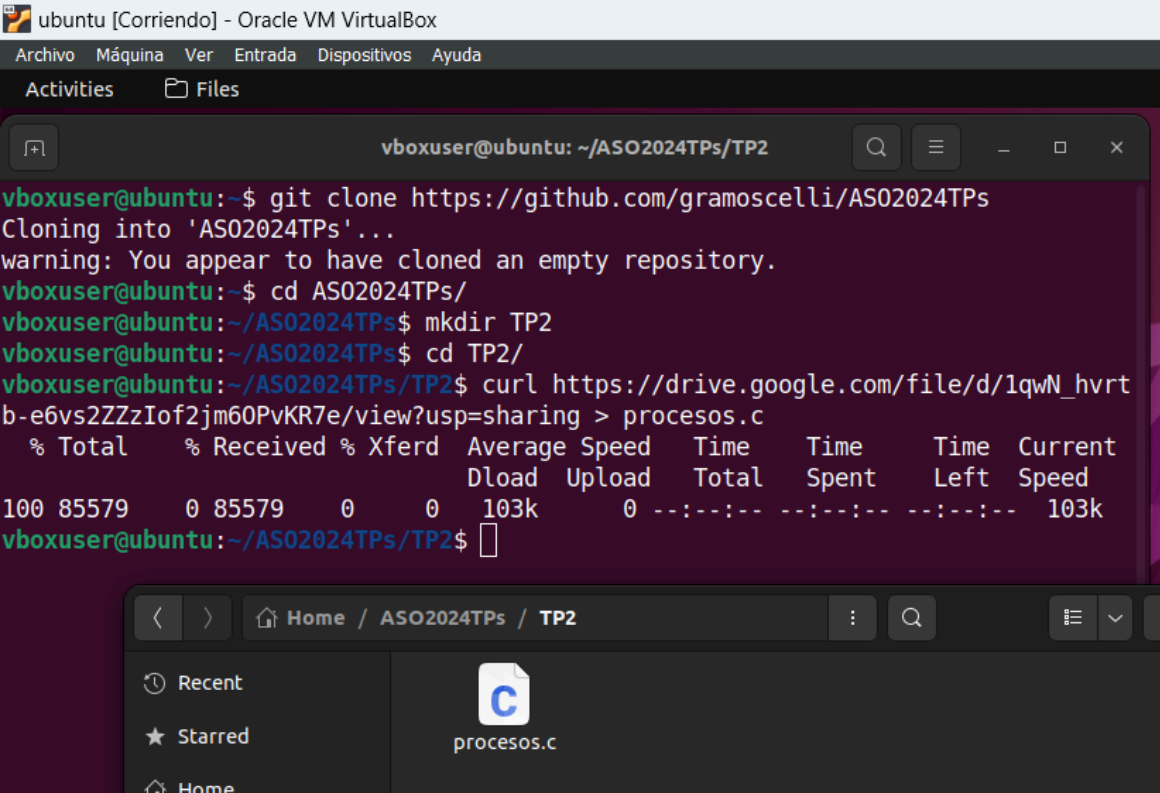
- Después entramos en la carpeta usando el comando cd:

```
cd TP2
```

- Ahora ya estamos dentro de la carpeta TP2, descargamos allí el archivo **procesos.c**. Esto lo podemos hacer usando el comando:

```
curl https://drive.google.com/file/d/1qwN_hvrtb-e6vs2ZZzIof2jm60PvKR7e/view?usp=sharing > procesos.c
```

- Lo hecho hasta ahora sería esto:



The screenshot shows a terminal window titled 'ubuntu [Corriendo] - Oracle VM VirtualBox'. The user is logged in as 'vboxuser@ubuntu'. The terminal output shows the following commands and their results:

```
vboxuser@ubuntu:~$ git clone https://github.com/gramoscelli/ASO2024TPs
Cloning into 'ASO2024TPs'...
warning: You appear to have cloned an empty repository.
vboxuser@ubuntu:~$ cd ASO2024TPs/
vboxuser@ubuntu:~/ASO2024TPs$ mkdir TP2
vboxuser@ubuntu:~/ASO2024TPs$ cd TP2/
vboxuser@ubuntu:~/ASO2024TPs/TP2$ curl https://drive.google.com/file/d/1qwN_hvrtb-e6vs2ZZzIof2jm60PvKR7e/view?usp=sharing > procesos.c
```

The output of the curl command shows a progress bar and a table of statistics:

% Total	% Received	% Xferd	Average Speed	Time Dload	Time Upload	Time Total	Time Spent	Time Left	Current Speed
100	85579	0	85579	0	0	103k	0	----	103k

The terminal ends with the prompt 'vboxuser@ubuntu:~/ASO2024TPs/TP2\$'.

- En este punto, el repositorio local es distinto del repositorio remoto porque agregamos un archivo (**procesos.c**) al repositorio local. Para subir el cambio (e.g. el nuevo archivo) debemos hacer un “push”. Para esto usamos los comandos de git de abajo:

```
cd ..
git add .
git commit -m “primer cambio”
git push origin main
```

- Ahora el repositorio remoto y el local están “sincronizados”.

7- Compilado del programa procesos.c

Dentro del directorio identificamos el archivo procesos.c

Podemos ver su contenido y examinar su código fuente, abriendo dicho archivo con el VSCode, por ejemplo, si se desea

En la consola compilamos el archivo procesos.c haciendo:

gcc -o <nombre_archivodestino> procesos.c

8- Ejecución del binario

Lo ejecutamos con

./<nombre_archivodestino> &

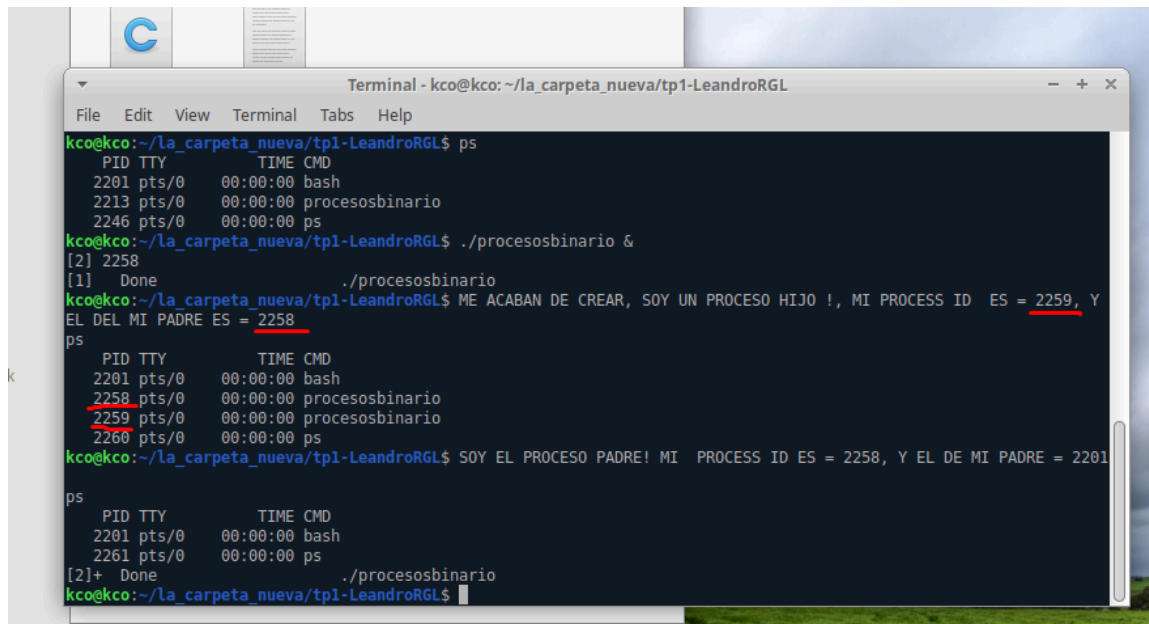
(notar el & al final, es para que ejecute el proceso en segundo plano)

Tip: **Ctrl + C** nos permite cortar o detener la ejecución de un comando

9- Listado de procesos corriendo

- Comando **ps** -> permite ver los procesos que están corriendo
- **Inmediatamente** ejecutamos el archivo anterior, ejecutamos el comando **ps**
- Podremos ver los procesos corriendo junto con su **PID** (Process ID)

(Las imágenes son de referencia y es muy posible que difieran en gran medida)



```
Terminal - kco@kco: ~/la_carpeta_nueva/tp1-LeandroRGL
File Edit View Terminal Tabs Help

kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ ps
  PID TTY          TIME CMD
  2201 pts/0    00:00:00 bash
  2213 pts/0    00:00:00 procesosbinario
  2246 pts/0    00:00:00 ps
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ ./procesosbinario &
[2] 2258
[1]  Done                  ./procesosbinario
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ ME ACABAN DE CREAR, SOY UN PROCESO HIJO !, MI PROCESS ID ES = 2259, Y
EL DEL MI PADRE ES = 2258
ps
  PID TTY          TIME CMD
  2201 pts/0    00:00:00 bash
  2258 pts/0    00:00:00 procesosbinario
  2259 pts/0    00:00:00 procesosbinario
  2260 pts/0    00:00:00 ps
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ SOY EL PROCESO PADRE! MI  PROCESS ID ES = 2258, Y EL DE MI PADRE = 2201
ps
  PID TTY          TIME CMD
  2201 pts/0    00:00:00 bash
  2261 pts/0    00:00:00 ps
[2]++ Done                  ./procesosbinario
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$
```

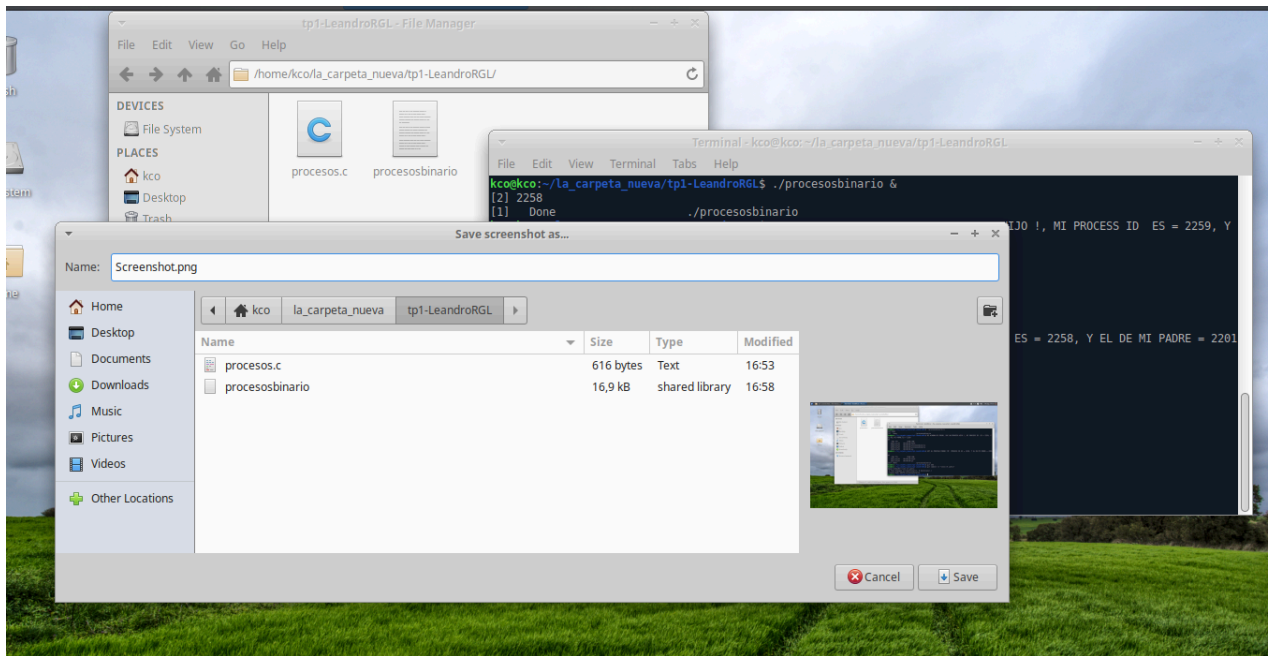
10- Captura de pantalla

Presionamos la tecla “print screen”

Debemos guardar la imagen capturada dentro de la carpeta del TP2, para que luego poder ser trackeada por Git

Según la configuración de cada S.O., la imagen capturada podría quedar almacenada en la carpeta usuario/imagenes

(Las imágenes son de referencia y es muy posible que difieran en gran medida)



11- GIT PUSH del TP2

- Para proceder a la entrega, se subirán los cambios de archivo, por caso el archivo de imagen capturada, a nuestro propio repositorio remoto, creado en los pasos anteriores
- Para ello debemos indicar a Git que utilizaremos un repositorio distinto al que clonamos en origen
- **git add .** -> trackea los archivos con cambio dentro de la carpeta
- **git commit -m "tarea finalizada"** -> realiza el commit de los cambios (está indicando una versión), notar que el comentario es obligatorio, y convenientemente debe indicar o dar una idea de que se logró en esta versión en cuestión
- Se ha realizado un commit de cambios de manera local, en el repo local

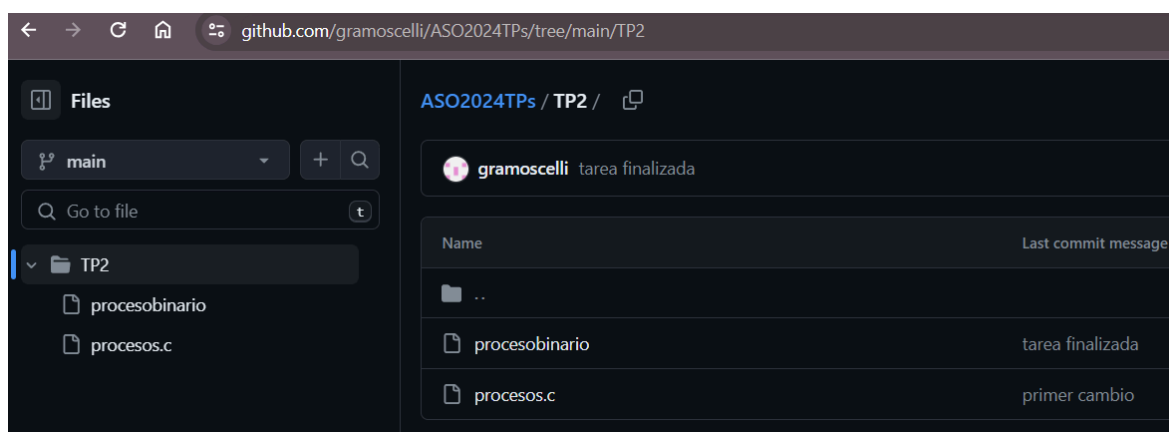
(Las imágenes son de referencia y es muy posible que difieran en gran medida)


```
Terminal - kco@kco: ~/la_carpeta_nueva/tp1-LeandroRGL
File Edit View Terminal Tabs Help

ps
  PID TTY          TIME CMD
  2201 pts/0    00:00:00 bash
  2258 pts/0    00:00:00 procesosbinario
  2259 pts/0    00:00:00 procesosbinario
  2260 pts/0    00:00:00 ps
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ SOY EL PROCESO PADRE! MI  PROCESS ID ES = 2258, Y EL DE MI PADRE = 2201

ps
  PID TTY          TIME CMD
  2201 pts/0    00:00:00 bash
  2261 pts/0    00:00:00 ps
[2]+  Done                  ./procesosbinario
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ git add .
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ git commit -m "listo el pollo"
[main b3f2af8] listo el pollo
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100755 procesosbinario
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ git add .
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$ git commit -m "listo el pollo con captura"
[main 94173f0] listo el pollo con captura
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Screenshot.png
kco@kco:~/la_carpeta_nueva/tp1-LeandroRGL$
```

- **git push origin main** -> lo “pushearé” (enviaré) a nuestro repo remoto en github
- Adicionalmente podemos dirigirnos a [https://github.com/\[nombre_usuario\]/ASO2023TP2](https://github.com/[nombre_usuario]/ASO2023TP2) y corroborar que nuestro repo remoto ha sido actualizado con los archivos de nuestro repo local:



12 Entrega del TP2

- Para la entrega del TP2, luego de haber hecho todos los pasos de arriba, debemos subir un archivo al aula virtual llamado `entrega.txt`. Dentro debe estar la dirección del repositorio de Github:

