Introducción a la Ingeniería Informática

Mini Proyecto C/C++

Sebastián Bórquez G.

sebastian.borquez@sansano.usm.cl

sborquez@alumnos.inf.utfsm.cl

Metas

- Aprender los comandos básicos de git
- Practicar con el lenguaje C++

Temas

- ¿Qué es git?
- Configuración de git y cuenta de gitlab.labcomp.cl
- Crear repositorio.
- Agregar miembro.
- Comandos básicos.
- Programar solución en grupos.

El problema

Simular los diccionarios de Python utilizando un arreglo de estructuras o una lista enlazada de estructuras.
Cada estructura representa al par key:value.
Crear un programa de consola, este recibe comandos y opera sobre nuestro único diccionario.

¿Qué es git?

- Git, es un sistema de control de versiones diseñado por Linus Torvalds.
- Nos facilita el desarrollo de software, especialmente cuando en el programa están trabajando varias personas. Además hace que sea más fácil la administración de las distintas versiones de cada producto desarrollado.
- Nuestro código queda almacenado en un repositorio en algún servidor, por lo todos los desarrolladores tienen acceso a la ultima versión de este.
- También existe la posibilidad de trabajar en diferentes características de forma paralela, y poder volver a versión anteriores si se cometen errores.

Configuración de git y cuenta de gitlab.labcomp.cl

- Ir a https://gitlab.labcomp.cl

  -Ingresar usuario y contraseña de informática

- Generamos una nueva ssh key:

```
$ ssh-keygen –t rsa –b 4096 –C «your_email@example»
```

- Mostrarmos nuestra public-key

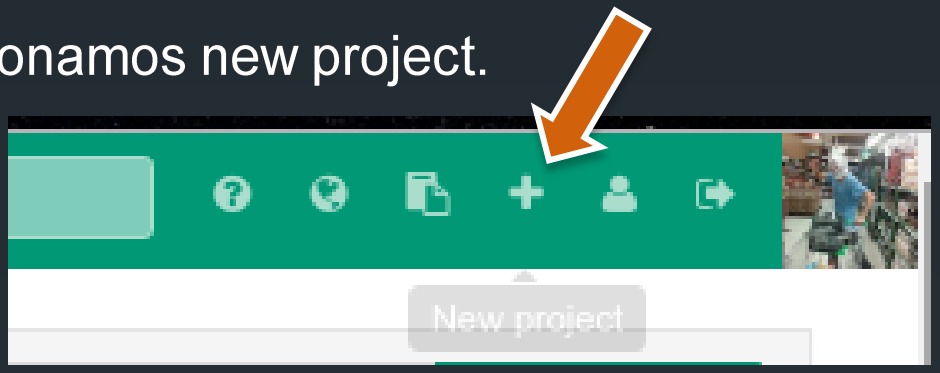
```
$ cat ~/.ssh/id_rsa.pub
```

- Accedemos a: profile settings -> ssh keys -> Add ssh key.
- Colocamos un titulo representativo.
- Configuramos el usuario.

```
$ git config –global user.name «your_name»
$ git config –global user.email «your_email»
```

Crear repositorio

- Presionamos new project.



- Ingresar nombre y descripción al nuevo repositorio.
- Marcar como public
- Create project

Agregar miembro

- Una vez dentro del proyecto:
- Settings->Members->Add members:

  Buscamos el usuario de nuestro colaborador, marcamos como master o developer.

Comandos básicos.

1. Clonar un repositorio:
   - En la gitlab.labcomp, copiamos el link de nuestro repositorio
   - Creamos una directorio para nuestros repositorios:

```
$ mkdir repos
$ cd repos
```

   - Una vez dentro del directorio usamos el comando «clone»

```
$ git clone git@gitlab.labcomp.cl:sbor.....git
```

2. Subir cambios:

```
$ git add <archivos>
$ git commit –m «comentario»
$ git push origin master
```

3. Actualizar directorio:

```
$ git pull
```

Comandos extras

```
$ git status          muestra los cambios de nuestro repositorio
$ git checkout --<archivo>          deshace los cambios locales
```

Ayudas

- Guia sencilla
  - http://rogerdudler.github.io/git-guide/index.es.html
- Comandos de git
  - https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf
- Generar ssh-key
  - https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/
  - https://gitlab.labcomp.cl/help/ssh/README.md

El problema

- Simular un diccionario de Python en C++.
- Nuestro programa debe manipular un único diccionario. Para esto debe ser capaz de manejar 4 comandos:
  - add <key>: agrega una llave al diccionario o modifica si existe.
  - remove <key>: elimina una llave y su valor del diccionario.
  - show <key>: muestra el valor de la llave, o nada si no existe.
  - exit: salir del programa.
- El proyecto debe estar constituido de 5 archivos:
  - main.cpp
  - comandos.cpp
  - comandos.hpp
  - Makefile
  - README.md
- Puede utilizarse un arreglo(estático) o una lista enlazada(dinámico).

Supuestos

- Utilizar una estructura que contenga una llave (string) y un valor (int) struct tupla;
- Suponga que el usuario sabe usar el programa y no ingresara datos incorrectos.
- Para el caso de arreglo:
  - Puede utilizar <vector>
  - Nunca habrá más de 100 llaves usándose.
  - Puede utilizas una llave especial que indique que el elemento del arreglo no está utilizándose.
- Para el caso de lista enlazada.
  - Usar una estructura nodo:

```
struct nodo{string llave; int value;struct nodo *sig};
                          o
struct nodo{struct tupla; struct nodo *sig}
```