



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Sintaxis y Semántica de los Lenguajes
Curso K2055
Esp. Ing. Pablo D. Mendez
Trabajo práctico Grupal Nro. 1

Generador de lenguajes (Trabajo práctico grupal)

Fecha de entrega: 13/06/2025

Introducción

En un lenguaje de programación existen diversos sub lenguajes regulares, por lo tanto, son una pieza fundamental. Se ha visto en clase distintas maneras de identificar cuando un lenguaje es regular.

El presente trabajo consiste en realizar un programa en lenguaje C que permita generar aleatoriamente palabras de un lenguaje regular a partir de una gramática ingresada por el usuario.

Desarrollo

El presente trabajo es grupal. Para ello se continúa profundizando la utilización de GitHub, ya que es la herramienta más utilizada para coordinar los elementos generados por un equipo de desarrolladores. Es fundamental, para la aprobación de este trabajo la creación de grupos no más de 5 personas y que todos ellos participen haciendo "Commits" en el repositorio.

1. Ingresar una gramática: esto quiere decir que se debe poder ingresar la cuatrupla: símbolos terminales, símbolos no terminales, producciones y axioma.
2. Validar si la gramática es regular o no: El programa no debe permitir el ingreso de gramáticas que no sean regulares.
3. Generar aleatoriamente palabras del lenguaje, cada vez que genera una palabra nueva debe mostrar el proceso de derivación que se realiza. Puede mostrar una derivación horizontal.

El programa debe estar prolijamente desarrollado, con código prolijo y modularizado (uso adecuado de subprogramas).

Las estructuras de datos utilizadas para la resolución quedan a criterio del equipo. No obstante, deben ser explicadas con detalle en el informe del trabajo.

Recomendaciones

- Investigue el uso de las funciones rand y srand para generar aleatoriedad. La aleatoriedad de la generación es un punto fundamental para la aprobación del trabajo.
- Crear un subprograma para el reemplazo de subcadenas, de modo de poder ir reemplazando los no terminales que van apareciendo en la cadena de derivación.

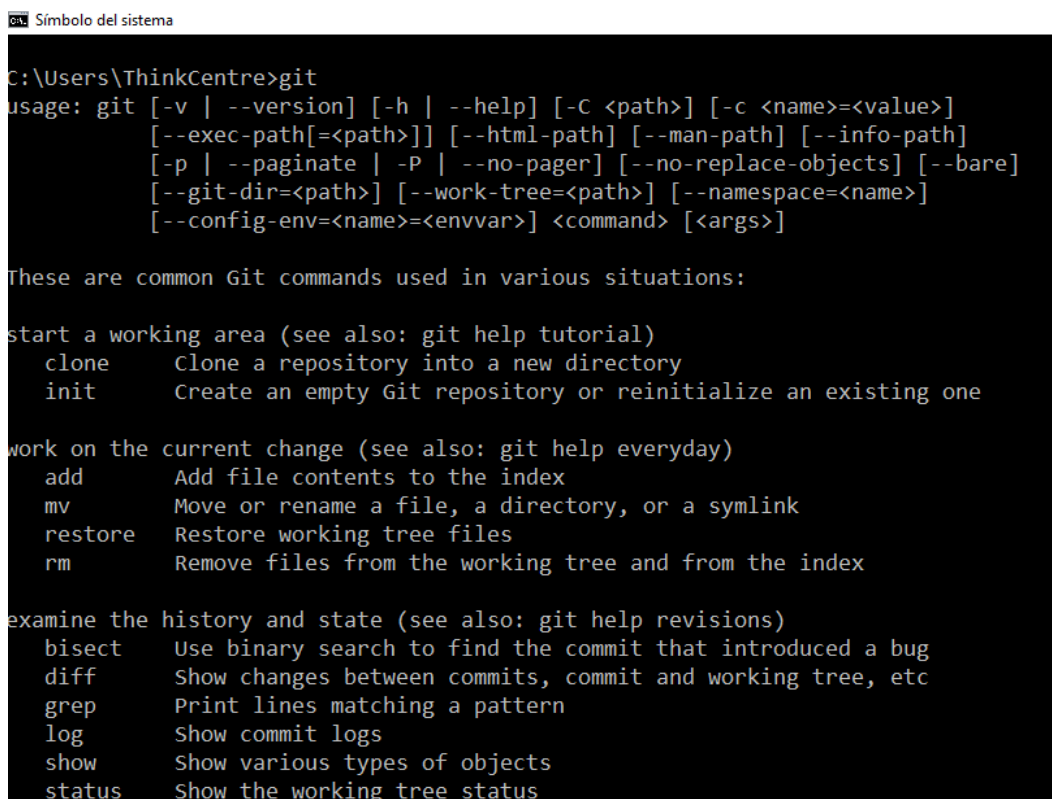
Instalación de Git y creación de repositorio local

En el trabajo anterior, se ha utilizado gitHub prácticamente como una carpeta en la nube, como un drive. En el presente trabajo, se utilizarán funcionalidades de gitHub imprescindibles para el trabajo en equipo en cualquier proyecto de software.

En este documento se muestra como utilizar los comandos básicos de git desde la consola. Ud. y su equipo pueden, sin embargo, utilizarlos desde cualquier IDE.

Descargue GIT para su sistema operativo desde la siguiente dirección <https://git-scm.com/downloads>

Una vez descargado puede verificar la correcta instalación desde la consola simplemente escribiendo el comando "git", si aparece la ayuda del comando como en la siguiente imagen, entonces git se ha instalado correctamente:



```
C:\Users\ThinkCentre>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  diff      Show changes between commits, commit and working tree, etc
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status
```

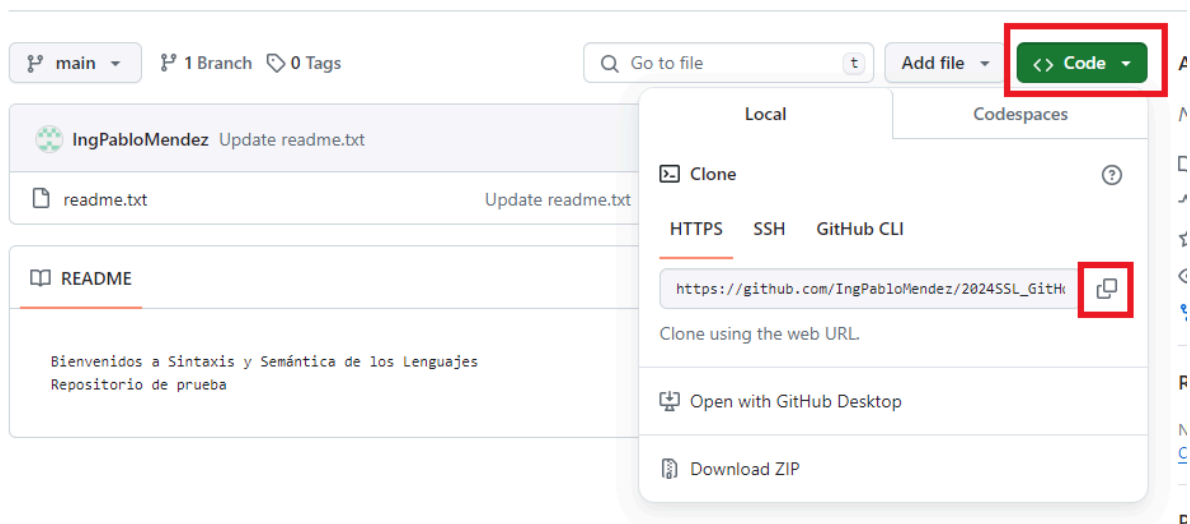
Para iniciar su repositorio utilice el comando "git clone", con el enlace al repositorio de su grupo, como se muestra en el siguiente ejemplo:

git clone https://github.com/IngPabloMendez/2024SSL_GitHow.git

Podrá ver la descarga de los archivos actuales del repositorio:

```
C:\Users\ThinkCentre\Desktop>git clone https://github.com/IngPabloMendez/2024SSL_GitHow.git
Cloning into '2024SSL_GitHow'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 1), reused 4 (delta 1), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
```

Puede obtener fácilmente la dirección del repositorio remoto en el botón que se muestra en la presentación del mismo:



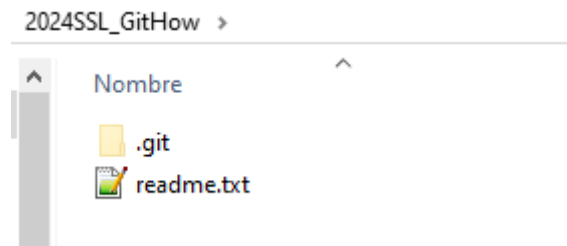
Ubicado dentro de la carpeta el repositorio, configure su usuario de github utilizando "git config", por ejemplo:

```
C:\Users\ThinkCentre\Desktop>cd 2024SSL_GitHow
C:\Users\ThinkCentre\Desktop\2024SSL_GitHow>git config user.email
pmendez@frba.utn.edu.ar
```

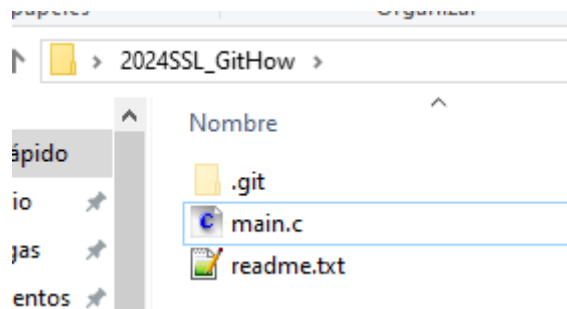
Luego de estos pasos, puede trabajar en su repositorio local normalmente.

Comandos básicos de uso intensivo:

- **git add**
Permite agregar un archivo en el repositorio. Si Ud. crea un nuevo archivo en el repositorio local, esto no quiere decir que luego se suba al repositorio remoto. Para hacer eso debe ejecutar git add. Por ejemplo, si el estado actual de la carpeta local fuera el siguiente:



Y luego el usuario agrega un archivo local, main.c:



para que este archivo se suba al repositorio remoto se debe ejecutar:

```
C:\Users\ThinkCentre\Desktop\2024SSL_GitHow>git add main.c
```

- git commit

Para confirmar cambios realizados o archivos agregados en un al repositorio, se debe ejecutar el commando commit. Esto no impacta directamente en el servidor, sino que genera un "buffer" de elementos a actualizar:

```
git commit -a -m "mensaje del usuario"
```

Por ejemplo para confirmar el archivo agregado en el punto anterior:

```
C:\Users\ThinkCentre\Desktop\2024SSL_GitHow>git commit -m "Se agrega el archivo main.c"
[main e9d5ce9] Se agrega el archivo main.c
1 file changed, 4 insertions(+)
create mode 100644 main.c
```

- git push

Finalmente, para subir los cambios confirmados al repositorio, debe ejecutar un git push:

```
C:\Users\ThinkCentre\Desktop\2024SSL_GitHow>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 370 bytes | 61.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/IngPabloMendez/2024SSL_GitHow.git
e9d5ce9..99d0839  main -> main
```

- git pull

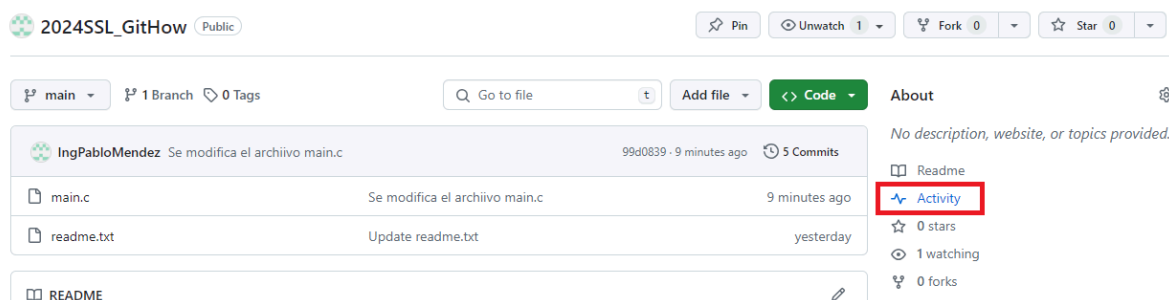
Siempre que comienza a trabajar (salvo la primera vez que inicia con un git clone) debe ejecutar un git pull para sincronizar su repositorio local con los últimos cambios realizados por los demás integrantes del equipo.

Si no hubiera cambios, git es muy claro:

```
C:\Users\ThinkCentre\Desktop\2024SSL_GitHow>git pull
Already up to date.
```

Importante: Siempre termine su sesión de trabajo con un commit seguido de un push. Y siempre comience su sesión de trabajo con un git pull (Excepto cuando hace git clone).

Otra ventaja de utilizar git es que los cambios son trazables, y siempre existe la posibilidad de volver a una versión anterior. Si Ud. quiere ver todos los commits del equipo puede hacerlo desde github:



2024SSL_GitHow (Public)

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

About

No description, website, or topics provided.

Readme Activity

0 stars 1 watching 0 forks

IngPabloMendez Se modifica el archivo main.c 99d0839 · 9 minutes ago 5 Commits

File	Commit Message	Time
main.c	Se modifica el archivo main.c	9 minutes ago
readme.txt	Update readme.txt	yesterday

README

Condiciones de entrega

1. Forme un equipo de hasta 5 personas (sin excepción). Informe el grupo a la cátedra en la planilla que corresponde.
2. La entrega del trabajo se debe realizar por GitHub, sin embargo, esta vez el repositorio grupal será creado por la cátedra y se invitará a los integrantes del grupo como colaboradores. Deben agregar los archivos dentro de la carpeta TP1 del repositorio, en donde deben estar los archivos .c y el informe del trabajo en formato PDF. Es importante que todos los integrantes demuestren participación en el trabajo.
3. El 16/6 se realizará entrega. El trabajo debe estar subido al repositorio para esa fecha.