Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Estudios de Postgrado
Maestría en ingeniería con especialidad en Ciencias de la Computación
Primer trimestre
Curso: Fundamentos de programación y scripting
Nombre: Miguel Antonio Orellana Ruíz

HOMEWORK 1

- **¿Qué es GIT?**

  GIT es un sistema de control de versiones distribuido, que se utiliza para el seguimiento y control de los cambios en proyectos de software.
  Inicialmente se creó para ayudar en el desarrollo de kernel de Linux, pero se convirtió en una herramienta indispensable para cualquier encargado de realizar proyectos de programación.

  Utilizando GIT los desarrolladores también pueden trabajar de manera colaborativa y coordinar los cambios realizados por varias personas en un proyecto determinado. Permite realizar un seguimiento de las diferentes versiones de los archivos, trabajar en diferentes ramas de un proyecto de manera simultánea y fusionar las diferentes versiones para crear una única versión final.

  Además, GIT ofrece una amplia gama de herramientas y características para facilitar el desarrollo de software, como el control de acceso, las etiquetas, las bifurcaciones, la reversión de cambios y la gestión de conflictos. Es una herramienta muy popular y ampliamente utilizada en la industria de la programación.

- **Control de versiones con GIT**

  El control de versiones con GIT es una forma de administrar el código fuente y el contenido de un proyecto de software, permitiendo a los desarrolladores trabajar juntos de manera efectiva y coordinada.

  En GIT, cada cambio realizado a un archivo se registra en un historial de cambios, lo que permite a los desarrolladores realizar un seguimiento de las diferentes versiones de los archivos y restaurar versiones anteriores si es necesario. Además, los desarrolladores pueden trabajar en diferentes ramas del proyecto de manera simultánea, lo que les permite experimentar y probar nuevas ideas sin afectar la versión principal del proyecto.

  Para comenzar a usar GIT, se debe inicializar un repositorio GIT en el directorio del proyecto. Los archivos y carpetas en el repositorio se agregan al área de

preparación (staging area) y luego se confirman (commit) los cambios en un repositorio local. Los cambios pueden ser sincronizados y compartidos con otros desarrolladores a través de un repositorio remoto, como GitHub o GitLab.

GIT también ofrece una amplia gama de herramientas y características para facilitar la administración del código, como la gestión de conflictos, las ramas, la reversión de cambios, la fusión de ramas y el etiquetado de versiones. Estas herramientas ayudan a mantener el código limpio y organizado, y facilitan el trabajo en equipo en proyectos de software.

- **Estados de un archivo en GIT**

  En GIT, los archivos de un repositorio pueden estar en uno de los tres estados principales:

  - **Modificado (modified):** Cuando se realiza algún cambio en un archivo existente del repositorio, el archivo entra en estado "modificado". Esto significa que se ha realizado alguna modificación en el contenido del archivo, pero todavía no se ha confirmado (commit) ese cambio en el repositorio.

  - **Preparado (staged):** Una vez que se ha realizado una modificación en un archivo y se desea guardar ese cambio en el repositorio, se debe agregar el archivo al área de preparación (staging area). El área de preparación es un área intermedia entre los archivos modificados y el repositorio, donde se preparan los cambios antes de confirmarlos. Cuando se agrega un archivo al área de preparación, se dice que está en estado "preparado".

  - **Confirmado (committed):** después de preparar los cambios en el área de preparación, se pueden confirmar (commit) en el repositorio. Cuando se confirma un archivo, se guarda el estado actual del archivo en el repositorio y se registra un mensaje que describe los cambios realizados. Después de la confirmación, el archivo entra en estado "confirmado".

  Además de estos tres estados principales, también hay un estado adicional llamado "*ignorado" (ignored),* que se aplica a los archivos que no deben ser rastreados por Git y que no deben incluirse en el repositorio. Los archivos ignorados no aparecerán en la lista de archivos modificados, y Git los ignorará cuando se realice una confirmación en el repositorio.

- **Como se configura un repositorio**

  Para configurar un repositorio en GIT, se pueden seguir los siguientes pasos:

  - **Inicializar un repositorio vacío:** Primero, es necesario inicializar un repositorio vacío en la carpeta donde se encuentran los archivos que se

desean agregar al control de versiones. Para hacerlo, se puede utilizar el comando git init en la línea de comandos, mientras se ubica en la carpeta del proyecto.

- **Agregar los archivos al repositorio:** Una vez inicializado el repositorio, se pueden agregar los archivos que se desean incluir en el control de versiones con el comando git add <nombre del archivo> o git add . para agregar todos los archivos en la carpeta de trabajo.

- **Confirmar los cambios:** Una vez que se han agregado los archivos al área de preparación (staging area), es necesario confirmar los cambios utilizando el comando git commit -m "mensaje de confirmación". Este comando crea una confirmación en el historial del repositorio que registra los cambios realizados en los archivos.

- **Configurar la conexión remota (opcional):** Si se desea compartir el repositorio con otros desarrolladores o almacenarlo en un servidor remoto, es necesario configurar la conexión remota. Para ello, se puede utilizar el comando git remote add <nombre del repositorio> <URL del repositorio> para agregar un repositorio remoto.

- **Empujar los cambios (opcional):** Si se ha configurado una conexión remota, es necesario empujar los cambios locales al repositorio remoto utilizando el comando git push <nombre del repositorio> <nombre de la rama>. Esto enviará los cambios realizados en el repositorio local al repositorio remoto.

Estos son los pasos básicos para configurar un repositorio en Git. A medida que se trabaja con el repositorio, es importante aprender a utilizar otras herramientas y comandos de GIT para manejar el control de versiones y colaborar con otros desarrolladores de manera efectiva.

- **Comandos en GIT**

GIT es una herramienta muy poderosa que proporciona una gran cantidad de comandos para administrar el control de versiones de sus proyectos. A continuación, se describen algunos de los comandos más utilizados en Git:

- **git init:** Inicializa un nuevo repositorio de Git en un directorio.

- **git add <archivo>:** Agrega un archivo específico al área de preparación para la confirmación.

- **git add**: Aagrega todos los archivos en el directorio actual al área de preparación para la confirmación.

- **git commit -m "mensaje":** Realiza una confirmación con un mensaje descriptivo que indica los cambios realizados.

- **git status:** Muestra el estado actual del repositorio, incluidos los archivos agregados y modificados.

- **git log:** Muestra un registro completo de todas las confirmaciones realizadas en el repositorio.

- **git branch:** Muestra una lista de todas las ramas disponibles en el repositorio.

- **git branch <nombre-rama>:** Crea una nueva rama con el nombre especificado.

- **git checkout <nombre-rama>:** Cambia a la rama especificada.

- **git merge <nombre-rama>:** Fusiona la rama especificada con la rama actual.

- **git pull:** Descarga los cambios más recientes del repositorio remoto y los fusiona con el repositorio local.

- **git push:** Envía los cambios locales al repositorio remoto.

Estos son solo algunos de los comandos más utilizados en GIT. Hay muchos otros comandos disponibles para administrar el control de versiones de sus proyectos. Se recomienda revisar la documentación oficial de GIT para obtener más información sobre los comandos y su uso.