Universidad de La Laguna
E.T.S. Ingeniería Informática
Área: Lenguajes y Sistemas Informáticos

Lenguajes y Paradigmas de Programación
Práctica de laboratorio **#1**

1. Iniciar una sesión de trabajo en GNU-Linux.

2. Abra una terminal.

3. Muestre el árbol de directorios de su HOME (`tree`).

4. Compruebe si existe el directorio *.ssh* (`cd ~/.ssh`).

5. Si la respuesta es "No existe tal directorio" continue por el ejercicio 8.

6. Si la respuesta es afirmativa, cree un directorio con nombre *copia* dentro del directorio *.ssh* (`mkdir ~/.ssh/copia`).

7. Mueva la pareja de clave-pública clave-privada al directorio *copia* (`mv ~/.ssh/id_rsa* ~/.ssh/copia/`).

8. Genere una nueva pareja de clave-pública clave-privada en el directorio *.ssh* (`ssh-keygen -t rsa`). Para usar las opciones por defecto, a cada pregunta responda pulsando la tecla de retorno de carro.

9. Cree un directorio de trabajo para la asignatura "Lenguajes y Paradigma de Programación". (`mkdir LPP`)

10. ¿Cuáles son las funcionalidades del comando `git`?. (`man git`)

11. Configure `git` con el **nombre de usuario** de manera que pueda etiquetar de forma correcta las actualizaciones que este realice. (`git config --global user.name "Nombre Apellido"`)

12. Configure `git` con la dirección de correo electrónico para asociarla a las actualizaciones que se hagan en el repositorio git. (`git config --global user.email "aluXXXXXXXXXX@ull.edu.es"`)

13. `git` permite almacenar la configuración global de un usuario en el archivo `.gitconfig`. Este archivo se encuentra en el directorio HOME del usuario. `git` almacena el remitente y el nombre del autor de un cambio en cada registro en el repositorio. Esta información se puede almacenar en el fichero de configuración global de manera que no se solicite cada vez que se haga un registro. Muestre el contenido del fichero `.gitconfig` (`cat .gitconfig`)

14. Configure `git` de manera que no sea necesario introducir la contraseña cada vez que se hace una actualización. (`git config --global credential.helper cache`)

15. Configure `git` de manera que todos los cambios se empujen siempre en el repositorio `git`. (`git config --global push.default "matching"`)

16. Configure `git` de manera que se eviten los registros (*commits*) innecesarios.

    (`git config --global branch.autosetuprebase always`)

17. Muestre las configuraciones globales de `git`. (`git config --list`)

18. Sitúese en el **directorio** de la asignatura "Lenguajes y Paradigmas de Programación" esto es en el directorio *LPP* (`cd LPP`).

19. Muestre el contenido del directorio actual (`ls -la`).

20. Cree un nuevo directorio denominado *prct01* (`mkdir prct01`). Este será el **directorio de trabajo** durante la realización de esta práctica.

21. Sitúese en el directorio *prct01* y cree la estructura de directorios que le permita tener subcarpetas para el código y los documentos, es decir:

    - un subdirectorio *src*
    - un subdirectorio *docs*

22. Guarde el fichero PDF que contiene el enunciado de esta práctica en el directorio *docs*.

23. Situado en el directorio de trabajo, inicialícelo para que sea un repositorio `git`. (`git init`)

24. Compruebe que se crea el directorio `.git` (`ls -la`).

25. Cree contenidos en el directorio de trabajo. Ejecute los comandos:

```
touch test01
touch test02
touch src/src01
touch src/src02
ls > test01
```

26. Compruebe el contenido del directorio de trabajo (`ls -la`).

27. Antes de realizar un registro en un repositorio Git es necesario marcar qué cambios se deben registrar añadiendo los nuevos ficheros y los ficheros cambiados al `índice del repositorio git`, esto es, al área de preparación. Esto crea una instantánea de los ficheros afectados. Si después de la instantánea, se cambia uno de los ficheros antes de registrarlos, es necesario añadir el fichero de nuevo al índice para registrar los nuevos cambios. Añada todos los ficheros y subdirectorios del directorio actual al *índice del repositorio git*. (`git add .`)

28. Registre (*commit*) los cambios del índice en el repositorio git local.

    (`git commit -m "First commit"`)

29. Muestre el fichero con los registros realizados en el repositorio hasta el momento. (`git log`)

30. Modifique los ficheros del directorio de trabajo. (

```
echo "Hola desde el fichero test01" > test01
echo "Hola desde el fichero test02" > test02
```

)

31. Compruebe las diferencias entre los ficheros anteriores y los nuevos. (`git diff`)

32. Registre (*commit*) los cambios del índice en el repositorio git local. La opción `-a` permite registrar los cambios de los ficheros modificados, pero no añade ficheros nuevos automáticamente al índice.
    (`git commit -a -m "Diciendo hola"`)

33. Modifique los ficheros del directorio actual. (

```
echo "Adios desde el fichero test01" > test01
echo "Adios desde el fichero test02" > test02
```

)

34. Muestre el estado del repositorio git local, esto es, qué ficheros han cambiado, cuáles son nuevos y cuáles han sido borrados. (`git status`)

35. Muestre las diferencias entre los ficheros sin registrar y los del último registro. (`git diff`)

36. Añada los cambios al *índice del repositorio git* y regístrelos.
    (`git add . && git commit -m "Maaaas cambios - con un error sintáctico en el mensaje"`)

37. Muestre la historia de los distintos registros (*commits*) en la rama actual. (`git log`)

38. Arregle el error en el mensaje del último registro (*commit*) del apartado 36.
    (`git commit --amend -m "Más cambios - ahora sin errores"`)

39. Cree un fichero y póngalo bajo el control de versiones. (

```
touch sinsentido.txt
git add . && git commit -m "se ha creado un nuevo fichero sin sentido"
```

)

40. Elimine el fichero del directorio. (`rm sinsentido.txt`)

41. Añada los cambios al *índice del repositorio git* y regístrelos.
    (`git add . && git commit -m "se ha eliminado el fichero sinsentido.txt"`)
    ¿Qué sucede? ¿Por qué no funciona?

42. Añada los cambios al *índice del repositorio git* y regístrelos con la opción -A. Con esta opción se consigue borrar un fichero de la instantánea de `git`.
    (`git add -A . && git commit -m "se ha eliminado el fichero sinsentido.txt"`)

43. ¿Qué es `GitHub`?

44. Cree una cuenta en *GitHub*.

    a) Abra en el navegador el sitio de GitHub: http://github.com
    b) Pulse el botón verde que aparece en pantalla.
    c) Introduzca como Nombre de Usuario su aluXXXXXXXXX
    d) Introduzca su dirección de correo electrónico institucional: aluXXXXXXXXX@ull.edu.es
    e) Introduzca su contraseña (utilice la misma que la de su cuenta institucional)
    f) Pulse el botón verde para crear la cuenta.

45. Muestre por la consola la clave-pública que ha generado (`cat ~/.ssh/id_rsa.pub`)

46. Copie en el almacenamiento temporal la clave-pública. Para ello, márquela con el ratón y pulse las teclas `Ctrl+C`.

47. Añada su clave-pública a *GitHub*.

    a) En la barra de usuario, en la esquina superior derecha de la página, haga clik en el icono de "Configuración de la cuenta" (*Account Settings*).
    b) En la barra de opciones que aparece en a la izquierda haga clik en la etiqueta "Claves SSH" (*SSH Keys*).
    c) Haga clik en el botón "Add SSH key" que aparece a la derecha.
    d) En el campo de texto Título (*Title*)) escriba *Centro de Cálculo*
    e) En el área de texto Clave (*Key*)) pegue (Ctrl+V) la clave que copió en el ejercicio 46.
    f) Pulse el botón verde para crear la clave (*Add key*).

48. Cree un repositorio en *GitHub*

    a) En la barra de usuario, en la esquina superior derecha de la página, haga clik en el icono de "Crear un repositorio nuevo" (*Create a New Repo*).
    b) Introduzca el nombre `prct01`
    c) Seleccione que quiere hacer el repositorio público.
    d) NO seleccione la casilla de crear el fichero README.md.
    e) Pulse el botón para crear el repositorio (*Create repository*)

49. Cree un repositorio remoto con nombre corto *origin*
    (`git remote add origin git@github.com:aluXXXXXXXX/prct01.git`)

50. Modifique el fichero `test02` (`echo "Se ha añadido un repositorio remoto" > test02`)

51. Ponga los cambios bajo el control de versiones.
    (`git commit -a -m "Esta es una prueba para el nuevo original remoto"`)

52. Empuje los cambios en el repositorio remoto denominado *origin*.
    (`git push -u origin master`)

53. Muestre los detalles del repositorio remoto denominado *origin*. (`git remote show origin`)

54. Muestre los repositorios remotos que están definidos. (`git remote -v`)

55. Escriba la dirección del repositorio que ha creado en GitHub en la tarea habilitada en el campus virtual.

56. Cierre la sesión.