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

Lenguajes y Paradigmas de Programación
Práctica de laboratorio **#2**

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

2. Abra una terminal.

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

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

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

6. Cree un nuevo directorio denominado *prct02* (`mkdir prct02`). Este será el **directorio de trabajo** durante la realización de esta práctica.

7. Sitúese en el directorio *prct02* y clone el repositorio remoto que se le indica. (

   ```
    cd prct02
   git clone git@github.com:coromoto/CompilacionVSInterpretacion.git
   ```

)

8. Compruebe que aparecen en el directorio `CompilacionVSInterpretacion` los siguientes ficheros (`ls -la`).

   ```
   helloWorld.c    - Lenguaje C   / compilado
   helloWorld.cc   - Lenguaje C++ / compilado
   helloWorld.sh   - Lenguaje Bash / interpretado
   helloWorld.py   - Lenguaje Python / interpretado
   HelloWorld.java - Lenguaje Java / compilado e interpretado
   ```

   Cada uno de los ficheros contiene un programa que muestra por pantalla la frase "Hello World".

9. Compilación en C:

   a) Muestre el contenido del fichero `helloWorld.c` sin abrirlo (`cat`).
   b) Compile el fichero `helloWorld.c` con el comando `gcc -o helloWorldC helloWorld.c`
   c) Compruebe que aparece en el directorio actual el fichero `helloWorldC` (`ls -la`)
   d) Ejecute el programa que se ha compilado con el comando `./helloWorldC`

10. Compilación en C++:

    a) Muestre el contenido del fichero `helloWorld.cc` sin abrirlo (`cat`)
    b) Compile el fichero `helloWorld.cc` con el comando `g++ -o helloWorldCPP helloWorld.cc`

c) Compruebe que aparece en el directorio actual el fichero `helloWorldCPP` (`ls -la`)

d) Ejecute el programa que se ha compilado con el comando `./helloWorldCPP`

11. Interpretación en Bash:

    a) Muestre el contenido del fichero `helloWorld.sh` sin abrirlo (`cat`)

    b) Ejecute el programa con el comando `bash ./helloWorld.sh`

12. Interpretación en Python:

    a) Muestre el contenido del fichero `helloWorld.py` sin abrirlo (`cat`)

    b) Ejecute el programa con el comando `python ./helloWorld.py`

13. Compilación e interpretación en Java:

    a) Muestre el contenido del fichero `HelloWorld.java` sin abrirlo (`cat`)

    b) Compile el fichero `HelloWorld.java` con el comando `javac HelloWorld.java`

    c) Compruebe que aparece en el directorio actual el fichero `HelloWorld.class` (`ls -la`)

    d) Ejecute el programa que se ha generado con el comando `java HelloWorld`

14. ¿Cuál es la diferencia entre compilación e interpretación?

15. Cree un fichero de texto con nombre `helloWorld.rb` que contenga lo siguiente:
    (`puts 'Hello world'`)

16. Añada el fichero `helloWorld.rb` al control de versiones.
    (`git add helloWorld.rb`)

17. Añada los cambios al *índice del repositorio git* y confírmelos.
    (`git commit -m "Hola Mundo en Ruby"`)

18. Ejecute el programa con el comando `ruby ./helloWorld.rb`

19. ¿Qué permisos tiene el fichero `helloWorld.sh`? ¿Se puede ejecutar directamente? (`./helloWorld.sh`)

20. ¿Qué permisos tiene el fichero `helloWorld.rb`? ¿Se puede ejecutar directamente? (`./helloWorld.rb`)

21. ¿Cuál es la principal diferencia entre el contenido del fichero `helloWorld.sh` y `helloWorld.rb`? ¿Qué significado tiene la primera línea del fichero `helloWorld.sh`?

22. ¿En qué directorio está instalado el intérprete de *Ruby*? (`which ruby`)

23. Modifique el fichero `helloWorld.rb` para que sea ejecutable. Primero edite el fichero y añada la primera línea correspondiente y a continuación establezca los permisos adecuados. (`chmod u+x helloWorld.rb`)

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

25. Añada los cambios al *índice del repositorio git* y confírmelos.
    (`git commit -a -m "Ruby ejecutable"`)

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

27. Muestre la historia de los distintos *registros* (`commit`) en la rama actual. (`git log --pretty=oneline`)
    Copie el número de identificación del registro "Primeros ejemplos" (`Ctrl + C`)

28. Muestre los cambios de un registro (*commit*) concreto mediante su identificador.
    (`git show <commit_id>`)

29. Muestre los ficheros que han sido cambiados en un registro (*commit*) concreto mediante su identificador.
    (`git diff-tree --name-only -r <commit_id>`)

30. Muestre los registros (*commit*) para el fichero `helloWorld.rb`.
    (`git log helloWorld.rb`)

31. Muestre las diferencias de cada uno los registros (*commit*) para el fichero `helloWorld.rb`.
    (`git log -p helloWorld.rb`)

32. Muestre el autor y el identificador de registro (*commit*) de cada una de las líneas del fichero `helloWorld.c`.
    (`git blame helloWorld.c`)

33. Cree una rama para hacer pruebas. (`git branch pruebas`)

34. Sitúese en la rama de pruebas. (`git checkout pruebas`)

35. Muestre la rama activa. (`git branch`)

36. Modifique el fichero `helloWorld.rb` para que reciba un argumento desde la línea de comandos y muestre la frase `Hola Mundo` seguida del nombre especificado. (`puts "Hello World #{ARGV[0]}"`)

37. Añada los cambios al *índice del repositorio git* y confírmelos.
    (`git add . && git commit -m "Argumentos en la línea de comandos"`)

38. Sitúese en la rama maestra. (`git checkout master`)

39. Fusione la rama de prueba en la rama maestra. (`git merge pruebas`)

40. Elimine la rama de pruebas. (`git branch -d pruebas`)

41. Cree un repositorio en *GitHub* con el nombre `prct02`

42. Muestre los repositorios remotos. (`git remote -v`)

43. Cree un repositorio remoto con nombre corto *miremoto*
    (`git remote add miremoto git@github.com:aluXXXXXXXX/prct02.git`)

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

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

46. Cierre la sesión.